FAE: The Fluid Analogies Engine

A Dynamic, Hybrid Model of Perception and Mental Deliberation

Scott William Bolland BA (Hons)

A thesis submitted for the degree of Doctor of Philosophy

School of Information Technology and Electrical Engineering

The University of Queensland
Australia

December 2004

Statement of Originality

The work presented in this thesis, is, to the best of my knowledge and belief, original, except as acknowledged in the text, and the material has not been submitted, either in whole or in part, for a degree at this or any other university.

_____

Scott Bolland, BA (Hons)

Abstract

This thesis describes the Fluid Analogies Engine (FAE), a computational model aimed at exploring the general cognitive mechanisms underlying flexible perception and intelligent deliberation. The architecture of this model was inspired by the work of Douglas Hofstadter and the Fluid Analogies Research Group who have developed a general computational framework for modelling high-level cognitive tasks including planning, creativity and analogy-making (Hofstadter & FARG, 1995). In this approach, high-level intelligent behaviour emerges from the parallel interactions of a multitude of agents that compete with and support each other in manipulating flexible, context-sensitive mental representations.

In addition to the Fluid Analogy models, there have been a range of computational frameworks proposed for modelling cognition using a complex systems approach. Such examples include LEABRA (O'Reilly and Munakata, 2000), DUAL (Kokinov, 1994b) and 3Caps (Just, Carpenter and Hemphill, 1996). As with the Fluid Analogy Models, these approaches are limited however, in that they utilise algorithms or representations that are only appropriate for modelling either low or high-level cognition. For example, connectionist models are limited in that they cannot readily model the manipulation of complex symbol structures required for planning and reasoning, and the Fluid Analogy models are limited in that they do not posses mechanisms for processing the raw distributed information provided by the senses. As is argued within this thesis, both high and low levels of processing are not only crucial for intelligent behaviour, they are heavily co-dependent and inseparable. The aim of the present work is to devise a general framework for cognitive modelling that can equally capture a wide range of both high and low levels of processing using the same set of mechanisms.

In bridging the gap between high and low level cognition, FAE uses a hybrid architecture involving a connectionist production system. All processing in FAE is achieved through a dynamically constructed neural network capable of processing distributed subsymbolic information. The network itself is formed through the actions of symbolically defined productions that afford the manipulation of complex hierarchical representations. FAE's general architecture includes a user defined set of sensory modules (for processing raw distributed data), a working memory (for the processing of symbol structures), a semantic memory (storing important "facts" that are known to the system) and an episodic memory (that allows for self-watching during mental exploration). FAE differs from other models of high-level cognition in that it utilises representations and algorithms that are equally applicable to both high and low level perceptual tasks, creating a unifying framework for general cognitive modelling.

List of Publications

Below are a list of publications produced during the time of candidature. Highlighted (*) are those related to work contained within this thesis.

* Bolland, S. & Wiles, J. (1998). Towards a connectionist model of high-level perception: what Copycat can that current connectionism can't. In Downs, T., Frean, M., and Gallagher, M., (Eds), *Proceedings of the Ninth Australian Conference on Neural Networks*, pages 287-91.

* Bolland, S. & Wiles, J. (2000). Adapting Copycat to Context-Dependent Visual Object Recognition. In Davis, C., van Gelder, T., & Wales, R. (Eds) *Proceedings of the Fifth Biennial Australasian Cognitive Science Conference*.

Humphreys, M., Dennis, S., Maguire, A., Reynolds, K., Bolland, S., & Hughes, D. (2003). What you get out of memory depends on the question you ask. *Journal of Experimental Psychology: Learning, Memory, & Cognition*. Vol 29(5), Sep 2003, pp. 797-812

Humphreys, M., Tehan, G., O'Shea, A., & Bolland, S. (2000). Target similarity effects: Support for the parallel distributed processing assumptions. *Memory and Cognition*, 2000 Jul; Vol 28(5): 798-811.

Wiles, J., Shulz, R., Bolland, S., Tonkes, B., & Hallinan, J. (2001). Selection procedures for module discovery: Exploring evolutionary algorithms for cognitive science. In J. D. Moore & K. Stenning (Eds), *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates, 1124 - 1129.

Wiles, J., Schulz, R., Hallinan, J., Bolland, S. & Tonkes, B. (2001). Probing the persistent question marks. In L. Spector, E. Goodman, A. Wu, W. B. Langdon, H.M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon & E. Burke (Eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001).* San Francisco, CA: Morgan Kaufmann Publishers, 710 – 717.

Acknowledgements

The completion of this thesis would not have been possible without the assistance, support, guidance and encouragement of many individuals.

Firstly, I would like to thank my wife, Melanie, for her love and support throughout the writing process. Hopefully now my weekends, nights and holidays can be filled instead by more mutually rewarding and enjoyable experiences. I would also like to thank my parents for their ongoing support (both emotional and practical), and Melanie's family for their provision of great food, accommodation, and a stable working environment over numerous summer holidays.

I am also deeply indebted to Janet Wiles (my supervisor) and Mike Humphreys (my employer over many years), who originally and continually piqued my interest in Cognitive Science. For all their feedback, insights and encouragement throughout the years I am greatly appreciative.

I would like to thank my colleagues at the University of Queensland who have contributed to making this endeavour a most enjoyable as well as fulfilling experience. In particular, for their friendship, advice, idle banter and intellectual stimulation I would like to thank Mikael Boden, Marcus Gallagher, John Hawkins, Andrew Smith, Bradley Tonkes and James Watson.

Finally, I would like to thank the School of Information Technology and Electrical Engineering for providing me with a desk and computing facilities, as well as the Australian Commonwealth Government for financial support (in terms of an APA scholarship).

Contents

Chapter 1
Modelling the Mind

> *… a single system (mind) produces all aspects of behaviour. It is one mind that minds them all. Even if a mind has parts, modules, components, or whatever, they all mesh together to produce behaviour. Any bit of behaviour has causal tendrils that extend back through large parts of the total cognitive system… If a theory covers only one part or component, it flirts with trouble from the start.*
>
> *(Newell, 1990, p17)*

The human mind is the most powerful and flexible information processor that we know, capable of complex feats of recognition, problem solving, planning, creativity and insight. One of the greatest challenges of cognitive science is to model the processes underlying such flexible and intelligent behaviour. Although much research has been devoted to the computational modelling of specific brain regions and processes (such as perception, memory, reasoning and motor control), as quoted above, Newell (1990) argues that intelligent behaviour emerges from the synergistic interactions between such constituents. Unified theories of cognition (UTCs) are an attempt to address this issue, modelling intelligent cognition and behaviour by defining the global architecture of the mind and detailing the various components and how they interact.

1.1 General Frameworks for Cognitive Modelling

In the creation of both unified theories of cognition and models of individual cognitive processes, there has been substantial debate over the past 50 years as to the

appropriate form (or level of abstraction) that such models should take. During this period, two main competing frameworks for cognitive modelling have evolved: the symbolic and connectionist approaches. In the symbolic approach, intelligent deliberation is modelled as the goal-directed manipulation of symbols, whereas in connectionism, cognition is modelled at a subsymbolic level, with models consisting of simple computational units analogous to biological neurons. As discussed below, these two frameworks capture disparate aspects of human cognition, with hybrid approaches so far showing the most progress in modelling flexible, high-level cognition.

1.1.1 The Traditional Symbolic Approach

The "traditional" or symbolic approach to cognitive modelling assumes that the human brain can be classified as a *physical symbol system* (*PSS*) (Newell & Simon, 1976). PSSs by definition contain a set of discrete entities, called symbols, that can exist within expressions, called symbol structures. Within this approach it is assumed that both the state of the world as well as the internal state of an intelligent agent (such as the agent's long-term or working memory) can be expressed in terms of symbolic descriptions (such as "book2 *is_on* table1" or "John *loves* Mary"). When given a task to perform, PSSs can generate solutions through the use of symbolic rules that can modify, create, reproduce or destroy the existing symbol structures. For example, in playing a game such as chess, a set of rules can be used to propose valid moves, with another set of heuristics being used to compare the options. According to Newell's *Physical Symbol System Hypothesis*, this class of system has "the necessary and sufficient means for general intelligent action" (Newell & Simon, 1976).

The main strength of the symbolic approach is in its success at performing tasks that to the human observer require intelligent deliberation, such as solving the Tower of Hanoi problem, or performing logical reasoning (Laird et al. 1987; McCarthy, 1990). In such tasks, the current state as well as possible solutions can be readily expressed in terms of discrete symbol structures. For example, in the Tower of Hanoi problem, the current state, the target state, as well as the results of possible actions, can all be expressed in terms of the discrete positions of the various disks. A physical symbol

system can display intelligent behaviour by explicitly representing the problem state and using its rules to heuristically navigate through the state space until an appropriate solution is produced.

To date, all of the most popular unified theories of cognition incorporate aspects of the traditional approach, manipulating their "mental" representations through sets of symbolic IF-THEN rules (called *productions*). Such models include SOAR (Laird, Rosenbloom & Newell, 1986; Laird, Newell & Rosenbloom, 1987; Newell, 1990), ACT (Anderson, 1983, 1993; Anderson & Lebiere, 1998), EPIC (Meyer & Kieras, 1997), 3Caps (Just, Carpenter & Hemphill, 1996) and DUAL (Kokinov, 1994b).

1.1.2 Connectionism

In contrast to the traditional approach to cognitive modelling, the connectionist approach asserts that "intelligence emerges from the interactions of large numbers of simple processing units" (Rumelhart et al. 1986, p. ix). In this approach, information, rather than being represented in terms of a set of distinct concepts, is represented subsymbolically as distributed patterns of activity over a shared set of simple computational units that are analogous to biological neurons. Information flows between such units through weighted connections that are analogous to the synapses in their biological counterparts.

By modelling cognitive phenomena at a subsymbolic level, connectionism can naturally capture many crucial properties related to perception and action that are difficult to model using the traditional approach. For example, in human vision, information provided by the retina is encoded as a visual array, similar to the pixel array captured by common camcorders. In this example, each rod or cone provides only partial evidence for the presence of an entity in the real environment with there being no simple mapping between symbols and sensory input, thus proving a challenge for pure symbolic models. The connectionist framework, by contrast, supports mechanisms for processing such distributed and subsymbolic information. In fact, presented with natural scenes and simple learning algorithms, such networks have been shown to develop the same kind of representations found in the human visual cortex (Olshausen & Field, 1996; O'Reilly & Munakata, 2000).

Consistent with connectionist implementations, many researchers believe that a large part of intelligent behaviour stems from subsymbolic processing, rather than mental deliberation at the symbolic level. As stated by Franklin (2001, p93), "I doubt if physical symbol systems could ever implement in real time the mental activity of a basketball player during a fast break." Such scepticism of the traditional symbolic approach is prevalent in the robotics community where researches require algorithms and representations that are appropriate for perception and action (e.g., Maes, 1991; Brooks, 1990). As argued by Brooks (1990, p5) in his paper entitled "Elephants Don't Play Chess":

> "…problem solving behaviour, language, expert knowledge and application, and reasoning are all rather simple once the essence of being and reacting are available. That essence is the ability to move around in a dynamic environment, sensing the surroundings to a degree sufficient to achieve the necessary maintenance of life and reproduction. This part of intelligence is where evolution has concentrated its time – it is much harder. "

According to Brooks, the seemingly goal-directed behaviours associated with "being" and "reacting" emerge through the interactions of simpler non goal-directed behaviours. Many of these simple behaviours such as obstacle avoidance and random exploration (that are required by many animals in the search for food or a mate), can be implemented as a direct coupling between perception and action without the use of symbols, mental deliberation, or an explicit internal model of the external environment. Connectionism offers an appropriate methodology for modelling such subsymbolic processes that are crucial for intelligent action.

1.1.3 Limitations of the Traditional and Symbolic Approaches

Both the traditional and symbolic approaches to cognitive modelling are limited in their ability to capture the full spectrum of processes underlying flexible cognition. In particular, as the traditional approach represents information as discrete symbolic entities, it is ill-equipped to handle the raw distributed data supplied by the senses or to model the direct subsymbolic pathways leading from perception to action.

Connectionism, by contrast, has limitations in its ability to manipulate and represent the complex symbol structures required for higher-level behaviours such as planning, problem solving, reasoning and complex decision-making. According to O'Reilly and Munakata (2000, p379) "in symbolic models, the relative ease of chaining together sequences of operations and performing arbitrary symbol binding makes it much more straightforward to simulate higher-level cognition than in a neural network." In neural networks, such processes are "intrinsically more difficult to model" as they involve "the coordinated action of multiple more basic cognitive mechanisms" (O'Reilly and Munakata, 2000, p379).

1.2 The Interplay between Perception and Symbolic Cognition

Although connectionism and the traditional symbolic approach seem to adequately model orthogonal mental abilities (i.e., perception and symbolic cognition), many researchers argue that there is no such clear cut boundary between these processes. For example, it is clear that how a situation is perceived is often greatly influenced by many top-down factors such as our beliefs, our goals and the current context. For example, what features come to mind when thinking of DNA differ depending on whether it is being comparing to source code or to a zipper (Hofstadter, 1995, p180), just as a monkey's perception of a banana will differ depending on whether it is planning on how to reach it, or deciding on whether or not it is rotten (Brooks, 1990, p4). This view that perception is an active and task-dependent process is strongly supported by psychophysical evidence (Yarbus, 1967; Toppino, 2003; Nelson et al, 2004).

Capturing the interplay between perception and symbolic cognition is central to effectively modelling the mental deliberation associated with flexible problem-solving and reasoning. Despite the tenants of the traditional symbolic approach, it is evident that symbols in the mind necessarily have fuzzy and context-dependent boundaries. For example, as illustrated in figure 1.1, in recognizing messy writing, ambiguous letters can often be interpreted as different symbols depending on the context at a word level (Selfridge, 1955). This phenomenon also exists in the perception of

words, with homographs (such as “fly”) being interpreted in different ways depending on the sentence context (Simpson & Kreuger, 1991). In such examples, the definition of what it is to be categorised as a concept (e.g., the visual input associated with a letter label) is affected greatly by the context in which it appears. Such context-dependencies in symbolic processing are not only crucial for normal recognition under impoverished conditions, but are believed to be ubiquitous in thought, underlying such high-order abilities as analogical reasoning and creativity (Hofstadter, 1995; Mitchell, 1993). For example, in forming an analogy, correspondences between non-identical entities need to be made that are often based on the context in which the entities appear rather than on similarities in superficial attributes (Gentner, 1983). For example, in drawing an analogy between the solar system and the Rutherford atom, the sun and nucleus are mapped, not due to their physical similarities, but because of similarities in their context (i.e., they both have other entities revolving around them). Capturing this context-dependent nature of symbolic processing is an essential component of modelling flexible high-level cognition.

Figure 1.1 An example of context-dependencies in symbolic processing. In the above sentence the “e” in the first instance of “the” is visually identical to the “c” in “cat” although they are interpreted differently due to their context at a word and sentence level.

1.3 Hybrid Approaches to Cognitive Modelling

Because of the limitations inherent to both the connectionist and traditional approaches in capturing all levels of cognitive processing, most progress in modelling high-level intelligent behaviour has been made by hybrid systems that attempt to merge the representational power of the symbolic approach with the flexible and context-sensitive nature of connectionism. Such models include ACT (Anderson, 1983, 1993; Anderson & Lebiere, 1998), 3CAPS (Just, Carpenter, & Hemphill, 1996) and DUAL (Kokinov, 1994b). These models gain their strength by modelling

symbols as discrete nodes within spreading activation networks whose activations affect the rate, speed or probability of symbolic rule firing. As the activation of nodes spread to connected concepts, processing in these models is highly context-dependent. Such models have proven successful in modelling a range of medium-level cognitive tasks including serial recall, recognition memory, free recall, implicit memory, similarity judgements, word-sense disambiguation and aspects of analogical reasoning (Anderson, Bothell, Lebiere & Matessa, 1998, Kokinov, 1989; Kokinov, 1992; Kokinov, 1994a).

In capturing the mechanisms underlying flexible mental deliberation using a hybrid architecture, the greatest progress at modelling a range of high-level tasks has been achieved by a collection of models developed by the Fluid Analogies Research Group (Hofstadter & FARG, 1995). Such models include Numbo and Jumbo (Hofstadter, 1995), Copycat (Mitchell, 1993), Tabletop (French, 1995) and Metacat (Marshall, 1999). These models differ from ACT-R, 3CAPS and DUAL in their ability to emulate human performance on complex problem-solving tasks such as planning, creativity and analogy-making utilising psychologically plausible mechanisms to manipulate symbol structures in a flexible and context sensitive manner. These models and the mechanisms that they employ will be described in detail in chapter two.

1.4 Limitations of Current Frameworks for Cognitive Modelling

In modelling flexible intelligent behaviour, it is evident that both subsymbolic and symbolic processes are essential (as discussed in previous sections). In the current literature however, not only does there exist no computational model that captures the full power of the perceptual and reasoning abilities found in humans (and the necessary integration between them), but there exists no general theory that adequately explains how this can be achieved. For example, although connectionism asserts that intelligent cognition emerges from the interactions of simple processing units, there is no specific theory that describes how the complex symbol structures required for high-level processing can be represented or manipulated.

Some progress has been made in bridging the gap between subsymbolic and symbolic cognition in a subset of the Fluid Analogy models including Copycat (Mitchell, 1993), Tabletop (French, 1995) and Metacat (Marshall, 1999). These models were explicitly developed to explore the integration between perception and reasoning, and in particular, to propose mechanisms to model the flexible and context sensitive nature of concepts. As with DUAL (Kokinov, 1994b) and 3Caps (Just, Carpenter & Hemphill, 1996), the Fluid Analogy models assert that high-level cognition can be modelled through the parallel interactions of symbolic agents that compete with and support each other in the gradual formation of complex symbol structures. The Fluid Analogy Models differ from DUAL and 3Caps however, in that they explicitly model contextual sensitivity, allowing the semantic definition of concepts to change under various situations (see chapter two for details).

Although the Fluid Analogy Models have successfully modelled a range of high-level cognitive tasks such as planning and analogy-making, they suffer from three main limitations. Firstly, in these systems, information is represented using discrete symbols (albeit with graded levels of activity) that make the approach limited to modelling the processes underlying mental deliberation rather than the aspects of intelligence that emerge from the transformations that occur at the subsymbolic level. Secondly, the mechanisms employed to account for contextual sensitivity have not proven to generalise to larger domains (for a detailed discussion, see chapter two). In particular, issues with scalability were noted in the implementation of Tabletop (French, 1995), with the chosen mechanisms being discarded in the more ambitions Letter Spirit Project (Rehling, 2000). These models are also limited in that each of the different models explores different aspects of cognition, with there being no single model that integrates all the crucial ideas. Thus, unlike SOAR and ACT-R, the Fluid Analogy approach has never been instantiated within a general modelling framework that can be readily applied to new domains or embodied within an intelligent agent.

1.5 Project Aims

The purpose of the project detailed by this thesis is to propose and implement a new computational modelling framework that better captures the processes of, and

integration between, perception and symbolic deliberation. As the models proposed by the Fluid Analogy Research Group explore a range of mechanisms related to this issue, these models provide inspiration for the basic theories and algorithms employed. However, the current project is aimed at reducing the limitations imposed by these models by integrating ideas from connectionism (to allow for the processing of subsymbolic representations) and exploring alternative, more general mechanisms to account for contextual sensitivity. Unlike the Fluid Analogy models, it is aimed that the resulting model should also be task-independent and able to be readily applied to new domains. To aid the development of such a generic modelling framework, ideas from current UTCs (such as the formulation of production-like languages) will also be explored.

1.5.1 Evaluation Criteria

In assessing the strength of current and new modelling frameworks that are aimed at capturing the mechanisms underlying human intelligent thought and behaviour, a set of evaluation criteria is beneficial. One of the most prominent such sets of criteria has been developed by Newell (1980, 1990) in the form of 13 constraints to which unified theories of cognition should adhere. Such constraints range from the ability of the system to function in real time, through to the ability of the system to arise through evolution. Although such constraints are important, within Newell's set of criteria there is little that describes the actual mechanisms underpinning intelligent behaviour that can be used to aid the construction of new frameworks. In contrast, Mitchell (1993) has outlined sets of cognitive abilities that are required in human high-level perception and intelligence (such as the ability to "mentally construct a coherently structured whole out of initially unattached parts" and the ability to "explore many plausible avenues of possible interpretations while avoiding a search through a combinatorial explosion of implausible possibilities"). Such concrete lists greatly help guide and constrain the design of computational models that mimic human cognitive behaviour. Outlined below, in no particular order, is a list that attempts to unify the most important cognitive abilities underlying intelligent behaviour, spanning both high and low levels of cognitive processing. This list has been inspired by the

insights of the Fluid Analogies Research Group, Newell, connectionist researchers and the robotics community.

Property #1: The ability to process raw distributed information

The ability to process raw distributed information is crucial for agents embodied in real world environments as the senses deliver distributed information from which symbols cannot be directly inferred. The ability to process such low-level representations is also necessary for the more direct pathways between perception and action that are deemed essential for intelligent behaviour (see Brooks, 1990).

Property #2: The ability to select and attend to relevant aspects of the world

Attention is an important aspect of mental deliberation as humans and animals have limited cognitive resources that make it impossible to process all aspects of the environment simultaneously (Cherry, 1953; Broadbent, 1958; Yarbus, 1967). In overcoming this limitation, humans perform general problem solving as a serial exploration, attending to only one avenue at a time (Newell & Simon, 1972). In dealing with real-world domains in which there are almost limitless alternatives, modelling the processes underlying this ability to select and attend to only relevant options is evidently important.

Property #3: The ability to process information in a contextually sensitive manner

Contextual sensitivity is an essential and ubiquitous property of the mind, occurring at all levels of perception and reasoning. For example, at a low level, examples of featural co-dependencies are evident, as exemplified by visual illusions such as the vase/profile illusion and old woman/young girl illusion (Rubin, 1915; Hill, 1915). In these examples, several interpretations are possible at a low level (such as the figure/ground classification of different edges) but are resolved globally, with only one consistent interpretation being perceived at a time. According to the Hofstadter (Hofstadter & FARG, 1995), contextual sensitivity also plays a central role in high-level tasks such as analogical reasoning and creativity. Thus, the ability to process

information in a contextually sensitive manner is a prevalent and necessary property underlying flexible intelligent cognition.

Property #4: The ability to create and manipulate complex relational structures

According to many researchers, the ability to represent and manipulate complex relational structures is a central component underlying intelligent deliberation (e.g., Hofstadter, 1995; Newell, 1990; Anderson, 1983; Deacon, 1997). The manipulation of such structures is not only useful in explicitly exploring the search space during problem solving (Newell, 1990), but the process of cleaving representations at natural boundaries and regrouping them in to different orders is thought to lie deep at the heart of creativity, being essential for such tasks as music composition, creative writing and scientific discovery (Hofstadter, 1995, p99).

Property #5: The ability to exhibit rational, goal-directed behaviour

In unconstrained real world domains, there are often infinite possible actions that can be taken at each step. For example, in achieving the goal of writing a novel, there are a multitude of actions that a person could choose to perform, ranging from writing drafts of chapters through to less productive endeavours such as watching daytime television. According to Newell (1980, 1990) however, humans generally act in a rational manner, choosing actions that support the goals that they wish to achieve.

Apart from in overt behaviour, selecting actions that are most appropriate in satisfying current goals is also necessary for mental deliberation. As humans have limited cognitive resources it is often infeasible to explore all possible alternatives either temporally or in parallel (as described above). Using rational heuristics to select actions that are likely to succeed in achieving the current goal allows many alternatives to be explored without a combinatorial explosion of possibilities.

Property #6: The ability to perform self-watching and mental regulation

In the serial exploration of alternatives during mental deliberations, a form of self-watching is required to prevent searching the same alternative many times. According

to Newell and Simon (1972), in solving complex tasks, human subjects do not conduct a breath-first or depth-first search of the problem space, but have reference points that they jump back to in order to continue the search. From such points however, it is evident that a memory for what has been attempted before is stored, allowing subjects to evaluate new alternatives and remember the best alternative found so far. Thus, it is beneficial for cognitive architectures to encode a form of self-watching during exploration, that remembers and recalls what alternatives have been explored, and regulates the search to explore the new territory.

1.6 Thesis Overview

This thesis describes the development and evaluation of a new modelling framework called FAE (the Fluid Analogies Engine), that was designed explicitly to explore the issues of integrating perception and mental deliberation. FAE can be best described as a hybrid symbolic-connectionist architecture, combining ideas from the Fluid Analogies Research Group (*FARG*), current unified theories of cognition and connectionist models. FAE differs from previous models in its ability to manipulate both low-level distributed information and high-level symbolic structures using the same set of mechanisms. FAE is capable of displaying all six of the core cognitive abilities described above, and is able to emulate all the major models devised by the Fluid Analogies Research Group utilising a unified framework that can also be readily applied to new domains and low level perceptual tasks.

The first half of this thesis (chapters 2-4) describes the background context, motivations and proposed solutions to the construction of a generic modelling framework inspired by the work of FARG. Firstly, to provide the motivating background of this project, the models proposed by FARG are overviewed (see chapter two). Within this section, the essential features that are considered to be useful in constructing a general framework are distilled, while highlighting the limitations of the current FARG models. Other cognitive models (both symbolic and connectionist) are then described (in chapter three), providing inspiration in overcoming the various limitations of the FARG approach. The Fluid Analogies

Engine is then described, providing details and motivations for the general architecture and algorithms employed (see chapter four).

The remaining chapters of this thesis (chapters 5-8) detail simulations that demonstrate FAE's capacity to display each of the six cognitive abilities described above. Firstly, visual word recognition is used to demonstrate FAE's ability to process distributed representations and display local contextual effects (see chapter five). Secondly, the ability of FAE to perform generic problem solving is explored, requiring the use of rational behaviour, hierarchical symbolic representations and self-watching (see chapter six). These processes are further explored within an anagram formation task that focuses on the process of creativity, using statistical regularities to help shape novel solutions (a domain problematic for purely symbolic accounts of cognition) (see chapter seven). Lastly, the processes of problem solving, selective attention and contextual sensitivity are further explored in a model of analogical reasoning (see chapter eight). The final chapter of this thesis (chapter nine), includes a general evaluation and discussion of the approach, a comparison to previous work, and noted implications for future research.

Chapter 2

Fluid Analogy Models of High Level Perception

The computational models of high-level perception devised by the Fluid Analogies Research Group (FARG) capture a combination of essential aspects of high-level perception that are ignored by unified theories of cognition such as SOAR and ACT-R. Such aspects include the flexible and context-sensitive nature of concepts, the necessity to focus computational resources only on the important aspects of the situation, and the ability to self-watch and regulate mental resources. These aspects allow the FARG models to perform levels of analogy-making, problem-solving and creativity that cannot be readily emulated by current unified theories of cognition.

Although each of the fluid analogy (FA) models captures essential aspects of high-level perception and can perform complex tasks, they have not been unified into a consistent framework that can readily be applied to new tasks. Furthermore, as will be discussed in this chapter, as a whole, these models are lacking in two of the six essential abilities required for intelligent behaviour as discussed in chapter one: they do not support the processing of raw distributed information, and the mechanisms that they employ for contextual sensitivity do not generalise well to other domains.

The aim of this chapter is to distil out the main mechanisms underlying the FA models that may be of use in the implementation of a new cognitive modelling framework. To act as a conceptual starting point of this project, the main philosophies central to the FA approach are introduced, followed by an overview of the main models. The strengths and weaknesses of these models are then compared to the criteria outlined in

chapter one, highlighting the mechanisms that have proven useful in capturing these core cognitive abilities.

2.1 The Philosophy of FARG

According to Hofstadter (1995, p97), the general tenant behind the Fluid Analogy Models can be captured by the slogan "Cognition equals Recognition", reflecting the belief that all high-level perceptual processing can be viewed as a form of recognition (the equating of non-identical percepts at some level of abstraction). Hofstadter (1979) used the Bongard problems (Bongard, 1970) to illustrate this point and to speculate on the processes involved. In the Bongard problems, two sets of six figures are given, with the aim of the task to find a rule that differentiates the sets (figure 2.1).

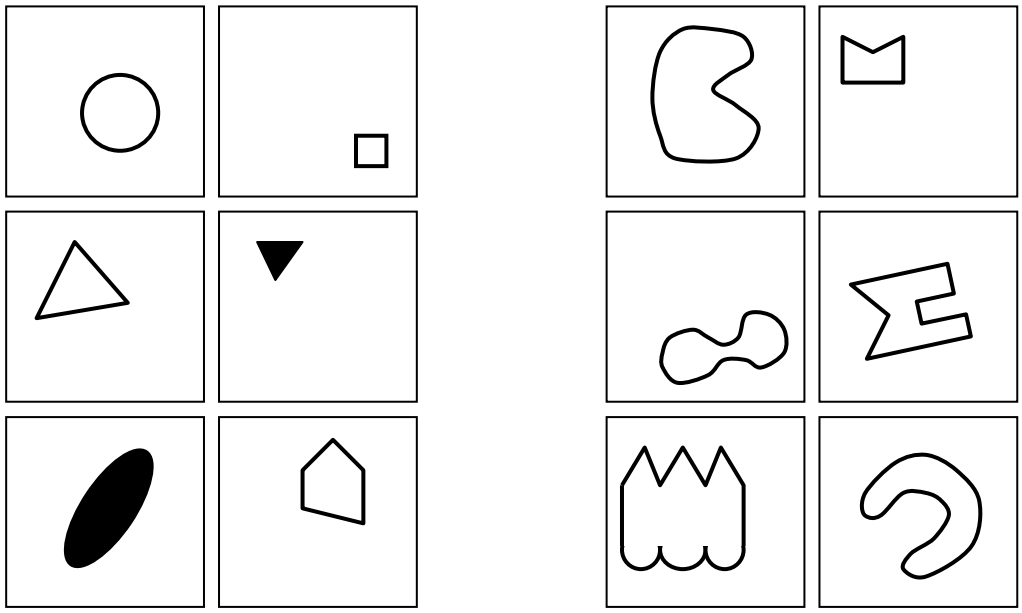


Figure 2.1 An example Bongard problem (redrawn from Bongard, 1970, p215). The aim of the task is to find a rule which segregates the six figures to the left from the six figures to the right.

In the Bongard problem above, a possible solution is that all the figures to the left are convex, whereas the figures to the right have sections that are concave. Such problems exemplify the ability of the cognitive system to recognise non-identical percepts (i.e., all the figures on the left or right) as being the same at some abstract level. According to Mitchell (1993, p3), this process is ubiquitous throughout the cognitive system, ranging from "basic and ordinary acts of recognition and

categorisation to rare and seemingly mystical feats of insight and creativity." In modelling the mechanisms underlying such processes, many of the FA systems model analogy-making as it is a task in which non-identical entities evidently need to be recognised as being the same at some abstract level. Such domains provide important insights into the issues involved in recognition, as many of the decisions that are made by an individual during this process are open to introspection. Furthermore, small analogy-making tasks can be readily formulated, computationally modelled in isolation, and compared to human subject performance for empirical validation.

2.2 Inspirations for the FARG Models

On the surface, the FA approach incorporates elements of the symbolic approach, manipulating symbol structures in order to find valid solutions to a given problem. As most interesting search spaces are too large to search exhaustively, one of the principle aims of FARG is to emulate psychologically plausible search strategies that reduce this search space. In fulfilling this goal, the FA models use a strategy called *the parallel terraced scan* that allocates more computational resources to the exploration of promising avenues. This parallel search strategy differs significantly from the approach taken by traditional UTCs such as SOAR and ACT-R (discussed in chapter three) that only allow a single action to be taken at a time. In contrast to such models, FARG views the brain as a self-organising system in which local stochastic parallel computations result in the emergence of a solution.

2.2.1 The Parallel Terraced Scan

In traditional approaches to searching large spaces, heuristics, such as means-end analysis have proven popular and effective in pruning large search spaces (e.g., Newell, 1990). However, according to Hofstadter, in many cases such approaches are too systematic and do not reflect the strategies actually employed by humans. Instead, Hofstadter states,

> "I felt what people do – or at least what I do – is more like an initial very shallow breadth-first scan followed by a bit of depth-first in a local area

> highlighted by the breadth-first scan, then resurfacing for more of a broad overview, then plunging back in perhaps somewhere else for a bit, and so on. In short, a constant interplay between episodes that tend toward the deep side and episodes that tend toward the broad side, with a constant willingness to jump out of any given mode and to try something different for a while…. There is also no intense resistance to looking again at an area already looked at once or even more times, because sometimes one comes back a second or third time with fresh new eyes." (Hofstadter, 1995, p34)

The Parallel Terraced Scan is a heuristic search strategy that was designed to emulate the above human approach to problem solving. The "parallel" in "Parallel Terraced Scan" actually refers to the fact that although a single main interpretation is perceived at a time, a mass of small, local (subconscious) searches are carried out in parallel that may redirect the main interpretation if proven promising. This process is likened to the search strategy employed by an ant colony looking for food, in which, although there is typically one main route taken by the whole colony, there are many small explorations made by various groups that can redirect the search.

Like an ant colony, the Parallel Terraced Scan regulates its resources so that more computational time is given to local searches that have proven promising, potentially shifting the main search into a new direction. As all searches are not treated equally, but are explored in relation to evaluated promise, the search is not exhaustive, but is pruned by exploring only likely avenues.

2.2.2 The Cell Metaphor

The general approach to representation formation in the FA models was inspired by viewing the brain as a self-organising system similar to the biological cell (Hofstadter & FARG, 1995). In the cell, as in the brain, there exists no central executive that oversees the construction of structures (molecular or mental). Rather, such structures are built through the parallel interaction of many small and local processes that compete with and support each other. In the cell, anabolic (bond-forming) or catabolic (bond-breaking) processes are carried out by enzymes that stochastically

move around the cell's cytoplasm. In the formation of a bond, if the enzyme runs into a molecule of the right kind (i.e. one that matches its active site), it will latch onto it and continue to travel the cytoplasm until a second molecule is found that matches its other site. Once this happens, the enzyme performs a chemical operation that bonds the two molecules into a larger structure. These larger structures can also be targeted by other enzymes that bond them together into high-order structures etc. Although such processes occur randomly throughout the cell there exist many important chemical pathways from which molecules are constructed efficiently and reliably. In such processes, no central executive is needed, but rather the process is self-regulatory, with the production of both complex molecules and enzymes being produced in accordance with the needs of the cell.

In accordance with the cell metaphor, the construction of representations in the FA models is conducted through the parallel activities of a large number of agents without the use of a central executive. These agents are called C*odelets* as each executes a small piece of program code that performs a specific operation. Codelets take the role of enzymes, bonding together the raw input to form coherent structured representations. In creating a new symbolic structure however, a number of Codelet actions are required, emulating a gradual flow of information. In the more advanced FA models such as Copycat (Mitchell, 1993), again like in the cell, the number of Codelets of each type can also be regulated by the system. In these models, this self-regulation allows for a natural interaction between bottom-up and top-down pressures that is required in the modelling of contextual sensitivity.

2.3 The FARG Models: An overview of the domains and architectures

The fluid analogy models were devised to function in small idealised microdomains. The microworlds were structured to capture and distil particular aspects of larger real-world domains so that the underlying phenomena could be studied in isolation in an understandable and controlled environment. To quote Hofstadter:

> "It is my firm belief that pattern perception, extrapolation, and generalization are the true crux of creativity and that one can come to an understanding of these fundamental cognitive processes only by modelling them in the most carefully designed and restricted microdomains" (Hofstadter, 1995, p86).

Each of the FARG models was devised to explore in detail a small range of phenomena relating to high-level perception. For this reasons, there are distinct differences in each of the implemented programs.

The next sections detail the various FARG models of high-level perception, describing the domains in which they function, the cognitive abilities that they were designed to capture, and their corresponding architectures. As the architecture of the more recent models build upon their predecessors, the simpler models are introduced first, focussing on the mechanisms that are shared with the later models.

2.3.1 Jumbo

Jumbo was the first model developed by FARG that incorporated the main architectural elements for problem solving utilised by the later models (Hofstadter, 1983). The aim of the project was to devise a computer simulation that used psychologically plausible mechanism to imitate human performance in a popular newspaper anagram game called "Jumble." The task of this game was to rearrange a string of letters (such as "h-s-o-t-g") to form an English word (such as "ghost"). In solving such a problem, humans use regularities such as likely letter and cluster co-occurrences to reduce the search space, instead of trying all 120 possible letter recombinations. This task is interesting in that it exemplifies central aspects of human intelligence and creativity, namely "the way in which we mentally juggle many little pieces and tentatively combine them into various bigger pieces in an attempt to come up with something novel, meaningful, and strong" (Hofstadter, 1995, p99). As judging the lexicality of formed word-like candidates is only used at the end of the anagram formation process (i.e. to decide whether or not a valid solution has been formed), this decision process was not considered central to the creative process and was not included in simulation. Rather, the aim of jumbo was to explore the

mechanisms underlying the creative process of forming word-like solutions, given information about how letters are typically arranged into consonant and vowel clusters, syllables and words.

The Architecture of Jumbo

Jumbo, like the rest of the FA models, contains two distinct memory systems: a working memory, in which the solutions are formulated (called the "Cytoplasm" in this model – in reference to the cell metaphor), and a long-term memory that contains the definitions of all concepts understood by the system. In Jumbo, the long term memory is called "the Chunkabet" as it stores the affinities of letters to appear and group in certain positions in the word. Using these affinity values, an overall fitness of letter combinations can be calculated, with the task of Jumbo to try to maximise the overall strength of the arranged sequence, creating a pseudoword that conforms well to the regularities found in English. The affinities used in Numbo were generated by Hofstadter through intuition (as he argued that it is through intuition that various letter combinations are preferred over others), with an integer strength value being assigned to potential vowel and consonant combinations that could appear at the beginning, middle or end of a syllable (figure 2.2).

sc: initial 2
sh: initial 8, final 8
sk: initial 4, final 4
s-ph: initial 1
oa: initial 2, middle 4

Figure 2.2: An example excerpt of the Chunkabet used by Jumbo (taken from Hofstadter, 1995, p104), showing the relative strength values of vowel and letter combinations occurring at the beginning, middle or end of a syllable.

Given the isolated initial letters, the aim of Jumbo is to gradually build up a hierarchical representation of potential word-like candidates, chunking the components at various levels including consonant and vowel clusters, syllables and words. This process is carried out using Codelets that randomly try to bond together molecules of different sizes (i.e. groups of letters), with each Codelet being targeted towards a specific entry of the Chunkabet. Due to the number of different ways in

which letters can be grouped, incompatibilities often arise in which several Codelets may be trying to process the same information in different ways (such as trying to bond *s* to form *s-t* or *s-h*). In such cases, only one such structure is permitted to form at a time (potentially destroying incompatible structures to do so), based on a probabilistic battle. Such battles are based upon the strength of the overall clusters, so that *s-h* may destroy an isolated *s-t* group, but is unlikely to beat the fully formed cluster *g-h-o-s-t*.

Jumbo and the Parallel Terraced Scan

In order to implement the parallel terraced scan, the formation of bonds in Jumbo are constructed in a number of steps, with each step being carried out by a different Codelet. These Codelets are stored in a data structure called the *Coderack* from which they are chosen to run one-at-a-time. By breaking down each task into a series of stages that can be interleaved with other processes, the Coderack emulates a form of parallel processing, allowing different tasks to run concurrently, potentially at different speeds. The decision of which Codelet to select to run is a probabilistic decision based upon its "urgency value" – in this case, the calculated affinity of the elements to be bonded. This probabilistic multitasking allows the more urgent processes to run at a faster rate and have more influence on a solution. In Jumbo, chunking of letters into discrete groups is achieved as a chain of three sequential steps, allowing many groups to form in parallel.

Selective Attention and Mental Regulation

In Jumbo, every new cluster, syllable or word that is formed is represented by a unique node in the Cytoplasm that is assigned its own set of properties. Such properties include its type (e.g. syllable or word) and *happiness*. The *happiness* of a node represents a measure of fitness of the structure and is a combination of internal and external factors. The *internal happiness* of a node represents how well the letter string matches the platonic ideal of the concept (which can be calculated from the *Chunkabet)*, whereas the *external happiness* is concerned with how incorporated it has become in other structures, with structures that have not been chunked into a

higher-order group being given a lower happiness rating. In such terms, it is Jumbo's ultimate goal to try to form a single "happy" chunk from the initial letters.

In helping regulate the computational resources of the system assigned to creating different structures in the Cytoplasm, a form of attention is implemented. Specifically, "unhappy" letters and groups "squeak" the loudest for attention, placing Codelets on the Coderack with higher urgency values. This process provides more attention to higher order structures as well as elements not contained in the coherent representations already formed. Thus, the attentive mechanism biases the parallel terraced scan to exploring ways of merging the currently unconnected structures, driving the system towards a coherent solution.

The Temperature

Generally speaking, Codelet actions support a decrease in the overall unhappiness of the system by creating hierarchical representation of increasing complexity. However, using this approach, it is possible to reach a dead-end solution from which no single operation will lead to a higher level of happiness. In such situations, backtracking is required in order to break up the suboptimal representation so that other alternative avenues may be explored. In Jumbo, backtracking is achieved through competing Codelets that can probabilistically destroy existing structures, as well as through the use of *dissolver Codelets* that randomly destroy unhappy structures. The release of dissolver Codelets is regulated by a global variable called the *Temperature* that reflects the overall happiness of the system. Dissolver Codelets are released (allowing backtracking to occur) when the system remains in a suboptimal state reflected by a high temperature. In contrast, once a good solution has been found (i.e. a single happy cluster has been found), the Temperature is set to "freezing" at which stage no dissolving Codelets are released.

The Temperature variable is a central component of most of the FARG models, having an effect on many processes. Generally speaking, apart from regulating the number of dissolver Codelets, the temperature is also used to alter the probabilistic decisions made by the system, becoming more deterministic as the temperature

lowers. This process is similar to simulated annealing, a popular computational search strategy regulated by a temperature value (Kirkpatrick, Gelatt, & Vecchi, 1983). In both algorithms, temperature is used to allow the system to escape from suboptimal solutions, but is lowered over time from an initially high value, to force the algorithm to settle into a solution. In the FARG models, rather than allowing the temperature to decay in a strict manner as in simulated annealing, the temperature is a function of the unhappiness of the system, resulting in initial random explorations, with a more directed search being conducted once a promising path has been found.

The Success and Limitations of Jumbo

One of the main strengths of the Jumbo model is in its ability to use letter co-occurrence information and the parallel terraced scan to exhibit rational-like behaviour. By assigning higher urgencies to Codelets that explore structures that more regularly appear in English words, these possibilities are created at a faster rate, resulting in the creation of words that reflect these regularities. If the initial biases lead to a dead-end, the probabilistic nature of processing and the Temperature regulated addition of dissolver Codelets, allows a form of backtracking to occur and for other avenues to be explored. Thus, Jumbo incorporates mechanisms that account for three of the core cognitive abilities required for intelligent behaviour outlined in chapter one: the ability to create complex hierarchical information, the ability to exhibit rational behaviour, and the ability to attend to only one main interpretation at a time.

The main limitation of the approach taken by Jumbo in modelling problem solving is in the lack of an episodic memory, resulting in an absence of memory for solutions already explored. Because of this omission, after backtracking, the dead-end solution may be re-explored a number of times, particularly if the system is heavily biased in this direction. As will be discussed, this limitation affects the scalability of the approach, hindering the performance of the later models such as Copycat in modelling problem-solving behaviour.

2.3.2 Numbo

Unlike Jumbo, which modelled aspects of human creativity, the aim of Numbo was to explore the mechanisms underlying human planning; choosing an appropriate set of operators in order to achieve a set goal. In the Numbo domain the aim is to "build" a given target number (a value between 1 and 150) from a given set of 5 numbers in the range of 1 to 25 (called bricks) using addition, subtraction and multiplication, with the restriction that each brick could be used at most once. Thus, this task differs from Jumbo in the sense that there is a strict goal that needs to be achieved, with the aim of the task to find a short and correct path. An example problem is given below:

Target: 87
Bricks: 8 3 9 10 7

In this problem possible solutions include (8*10)+7, and (9*7)+(8*3). However, of these solutions, only the first is generally given by human participants due to the saliency of certain operations over others (Hofstadter & FARG, 1995, p 133). It was the aim of Numbo to try to mimic the biased nature of this search, only entertaining ideas that are similar to those generated by humans. For this purpose, Numbo was given a long-term memory of facts about small numbers (such as 2*3=6) and landmark numbers that have well known decompositions (such as 100 = 20*5 or 10*10). From these facts, other closely related facts could be inferred, such as the fact that 19*5 will lead to a number around 100. However, such derivations are more costly on computation, and are therefore less likely to be utilised than well-learned combinations.

An interesting and important distinction from the Jumbo project is that in many of the problems, humans tended to solve them using a set of nested subgoals. As stated by Defays (Hofstadter & FARG, 1995, p133), in solving the above problem, participants generally state something like:

> "I see a 10 and an 8. The target can be decomposed as 80+7. I still need a 7. Oh, there is a 7! Therefore, the solution is 8*10+7."

Thus, the Numble domain represents a larger class of problems in which goals and subgoals are clearly defined, with a standard set of operators being used to negotiate a path to the solution.

Numbo's architecture is more complex than Jumbo's, sharing many important features with the later Fluid Analogy models such as Copycat. In particular, Numbo's long-term memory is represented by means of a semantic network that regulates the computational resources of the system. In this network, salient numbers, as well as multiplication and addition facts (such as the fact that 5*20=100) are represented by separate nodes. These nodes are connected by labelled links, denoting such relations as the similarity between 5 and 6, and that 100 is the multiplicative result in the stored fact that 5*20=100 (see figure 2.3).

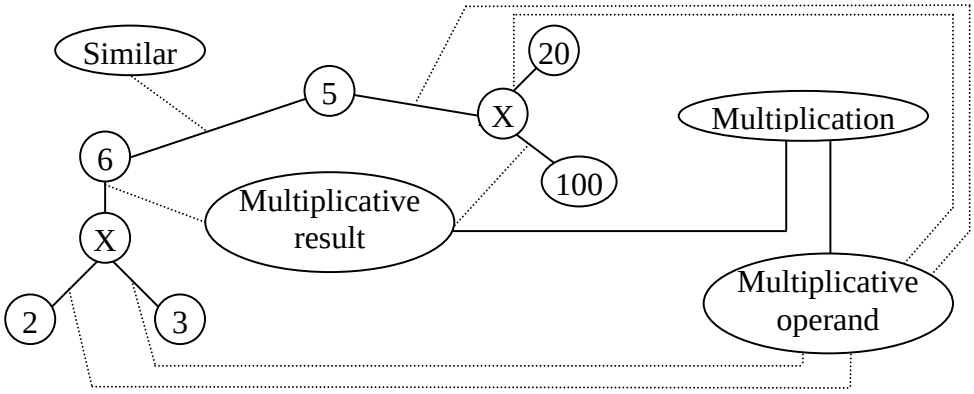


Figure 2.3: A subsection of Numbo's Long Term Memory. Numerical Facts are stored explicitly in terms of links between nodes. All links are also instances of concepts known to the network (adapted from Figure III-1 from Hofstadter, 1995, p136)

As stated, Numbo's long-term memory, as with the later FARG models, is used in the regulation of resources (i.e. influencing the number and speed of Codelets assigned to each task). In particular, when certain nodes in the network become active (such as any of the multiplication instances), Codelets will be spawned that actively try to build corresponding structures in working memory. Such activation is gained through spreading activation from concepts that have already been perceived in the workspace. For example, when the target is close to 100, and there are bricks with values of 5 and 20, the concept nodes "100", "5" and "20" would be highly active. These concepts would spread activation to the multiplication fact "5*20=100" that

will spawn Codelets that will try to build this structure in working memory. If bricks are found by the spawned Codelets with values of 5 and 20, a new object in the Workspace will be created (called a Block) with a value of 100 that combines these two bricks.

Subgoaling

In many cases and reflecting human performance, solutions in Numbo can often be solved through the use of nested subgoals. In Numbo, subgoaling is achieved by calculating the discrepancy between the current target and the various bricks and blocks present in the Workspace (a form of means-end analysis). If this discrepancy is small, Codelets can create a "secondary target" in the Workspace explicitly storing this value. For example, if the current goal is 87, and there exists a block with a value of 80, a secondary target of 7 may be created (see figure 2.4). This new structure activates additional facts in long term memory that may be beneficial in forming a solution. For example, if there are numbers in the Workspace (such as 3 and 4) that can be combined to achieve a value close to the new target (7), the corresponding facts will be highly activated (i.e, 3+4=7), spawning Codelets to explore this connection, leading to a solution.

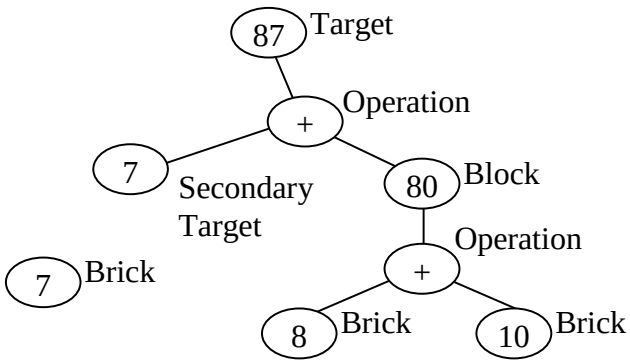


Figure 2.4: the creation of a Subgoal in Numbo. Once a Block is created that is close to the Target, the difference may be explicitly calculated, creating a secondary target that needs to be matched.

The Success and Limitations of Numbo

As a model of human cognition, one of Numbo's main strengths is in the ability to mimic human performance across a range of problems, occasionally preferring answers that incorporate well-learned facts over optimal solutions (Hofstadter and FARG, 1995). A second strength of the approach is that the basic mechanisms underlying solution formation are also employed by the later FA models such as Copycat, Tabletop and Metacat, thus displaying the generality of the approach to a number of different domains.

As Numbo uses the same set of mechanisms as the other FA models however, it suffers from the same limitations. Most importantly, Numbo, like Jumbo, lacks an episodic trace that could be used for more intelligent backtracking. Such a trace in planning is useful for exploring several different alternatives (such as a set of potential moves in chess), remembering and comparing the different solutions so that the best alternative can be chosen.

2.3.3 Copycat

Copycat (Mitchell, 1993) was the first of the FARG projects to explicitly model the process of conceptual slippage (the equating of non-identical entities). To study this phenomenon, a microdomain of letter-string analogy problems was used. In this domain, the system is given an initial transformation (such as *abc→abd*), and is required to be "a copycat" and to modify a target string (such as *ijk*) in the same manner. Such a problem requires the use of conceptual slippage, as in drawing an analogy across the strings, the letters in the target string (such as *k*) often need to be equated with non-identical letters in the source string (such as *c*). Rather than being based on letter category, such correspondences are often made due to the similarities in the context in which they appear (such as the fact that they both appear as the leftmost letter in the string, or appear as the highest letter in a group of alphabetic successors).

The Workspace

Copycat's working memory, called the *Workspace*, contains a set of symbolic data structures consistent with unified theories of cognition such as SOAR and ACT-R (see figure 2.5). Firstly, *objects* are represented as distinct entities that can be assigned a list of *properties*. In the case of Copycat, objects can either be individual letters or groups of adjacent letters that have been chunked into a single unit. The properties that objects can be assigned denote attributes such as spatial position within a string, letter category and length. Based upon related attributes, Codelets can build bonds between objects, denoting the perceived relationship. Such bonds include *successor* and *predecessor bonds* (such as a *rightgoing successor bond* between *a* and *b* in the string *abc*), as well as *sameness bonds* (denoting the identity relationship between the adjacent *j*s in the string *iijjkk*). Similar to bonds, *correspondences* can also be found between objects across domains based upon similar attributes (such as two letters being equated because they are the same letter category, or share the same spatial position in the different strings).

Apart from objects, properties, bonds and correspondences, Copycat uses two other structures in the formation of a solution: an *initial rule* and a *translated rule*. The initial rule is in the form "replace the X of Y by Z", allowing such transformations as "replace the letter category of the rightmost letter by its successor" and "replace the letter category of *c* by *d*" (both valid explanations of *abc→abd*). The translated rule in contrast, contains the rule to be directly applied to the target string to solve the analogy. Although the initial and translated rule may be identical, they may also contain conceptual slippages, reflecting the slippages found in the list of correspondences. For example, in the problem *abc→abd, cba→?,* a likely answer would be *dba*. In this solution, a mapping between the *c*'s across the strings is required, incorporating a slippage between the concepts *rightmost* and *leftmost* (descriptions assigned to the spatial positions of the two objects). Thus, if the initial transformation was described as the rule "replace the letter category of the rightmost letter by d", the transformed rule would be "replace the letter category of the leftmost letter by d", allowing the mentioned solution to occur.

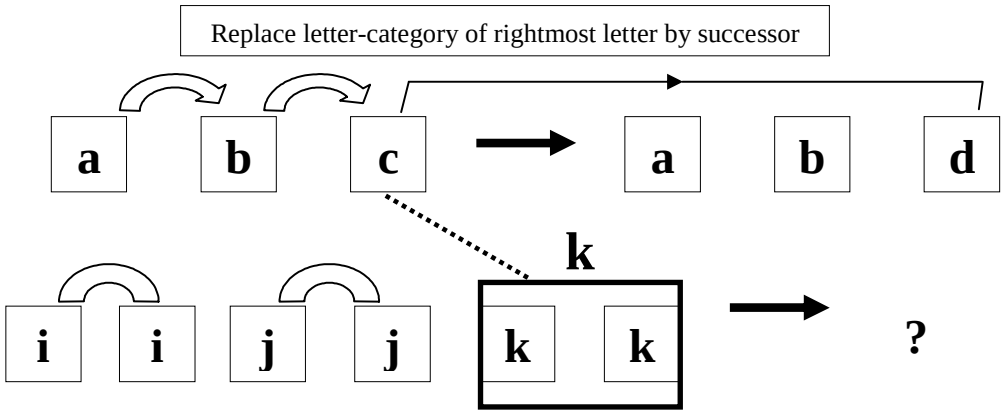


Figure 2.5. An example of Copycat's Workspace (adapted from Figure 3.3 from Mitchell, 1993, p41). This figure shows successor bonds between ab and bc, sameness bonds between identical letters, a sameness group containing the two ks and a correspondence between c and kk. A rule explaining the initial transformation is also given.

Solution Formation, the Slipnet and Conceptual Slippages

As with Jumbo and Numbo, in Copycat, problem solving occurs through the activities of large amounts of Codelets that perform a range of tasks at various speeds. Such tasks involve creating bonds between adjacent letters (such as creating a successor bond between *ab* in *abc*), chunking letters into higher-order groups that can also be assigned properties (such as treating *abc* as a group of successive letters), creating correspondences across strings, and creating rules and transformed rules. As with the other domains explored by FARG, representation formation in Copycat is complicated by the existence of a range of possible solutions. For example, in the problem *abc→abd, kji→?*, possible solutions include *kjd, kjh*, and *lji*, each of which require different groups, correspondences or rules to form.

During solution formation, one of the initial ambiguities that needs to be resolved in Copycat is the direction and type of bonds that are formed in the Workspace. In particular, the relationship between the letters in a string such as *ab* can be expressed as either a *successor bond* to the right, or a *predecessor bond* to the left, but not both at the same time. In solving the problem *abc→abd, ijk→?* however, all bonds should be of the same type and direction, requiring a source of pressure to aid the formation of a consistent interpretation. In Copycat, such pressure comes in three forms: the internal strength of structures, the formation of groups, and the activation levels of

concepts in the Slipnet. Firstly, the internal strength of bonds is increased with the number of similar bonds perceived in the same string, strengthening the structures, and making the alternatives less likely to win any probabilistic battle. Secondly, letters that have been bonded together (such as the letters in the string *abc*) can be chunked into a group (a *right successor group*, or *left predecessor group* in this case) that further strengthen the interpretation. Such groups lessen the attention paid to the underlying objects (lessening the pressure to form the alternative bonds), and also act as a higher-order structure that need to be destroyed in a probabilistic battle to form the alternative bond (further solidifying the underlying structure).

The third form of pressure to resolve ambiguities stems from the Slipnet (Copycat's long-term memory). In the Slipnet, there are nodes representing all the concepts known to the system, including bond types and directions. When an instance of the concept is built in the Workspace, the corresponding node becomes active, placing top-down pressure on the system (through the posting of specific top-down Codelets onto the Coderack) to try to build more bonds of this type. Thus, as soon as a successor bond or group is found in the Workspace, extra pressure is exerted to probabilistically prefer this interpretation. These three forms of pressure help Copycat to form globally consistent interpretations over time.

Apart from regulating top-down pressures, the Slipnet provides the main mechanism for allowing conceptual slippages to occur. In the Slipnet, potential conceptual slippages (such as between *leftmost* and *rightmost*) are encoded by explicit "slip links" between concepts (see figure 2.6). This system differs from other semantic networks in the fact that the length of these links (relating to how easy it is to make such a slippage) can be modified by the situation at hand. For example, if it were easy to generally slip between *rightmost* and *leftmost*, a potential solution to the problem *abc→abd, ijk→?* would be *hjk*, which is not an answer typically given by human subjects. Copycat regulates the occurrence of such context-dependent slippages, by modifying the length of Slip Links in relation to the activation of the concept representing the link type itself. For example, when the concept "opposite" becomes active in the Slipnet, the link between rightmost and leftmost will become shorter, allowing for this slippage to occur more freely. This particular situation will occur (i.e. opposite will become active) in problems such as *abc→abd, kji→?*, when the

system recognises that the initial strings are both successor groups in opposite directions. Once the correspondences between the two groups are made and the opposite node is activated, the leftmost and rightmost letters can become equated, leading to the solution *lji*.

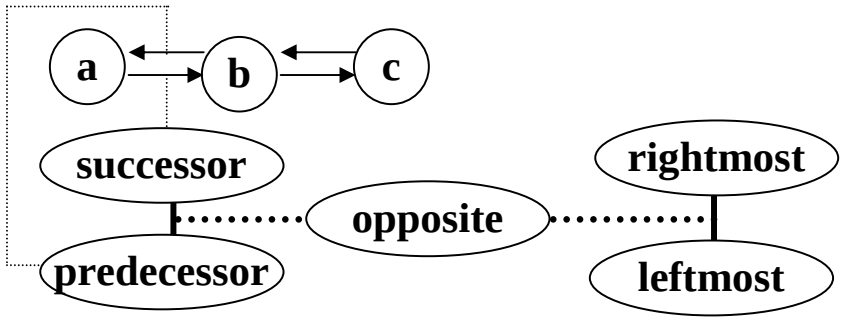


Figure 2.6. A small subsection of Copycat's Slipnet showing some of the more important concepts. Letters in the alphabet are linked through successor links in the forward direction and predecessor links in the reverse direction. The link types are themselves represented by nodes in the Slipnet. Concepts such as successor and predecessor are also connected through slip links, representing potential slippages. The length of the slip links are related to the activation levels of the link type node (i.e. "opposite" in this case).

The success and Limitations of Copycat

On a range of letter-string analogy problems in which there are multiple solutions, Copycat is able to produce the different answers provided by human subjects with approximately the same probability (Mitchell, 1993). One of the strengths of this model is that in formulating such answers, Copycat utilised a rich underlying knowledge of the domain (stored in the Slipnet). The use of such knowledge is in contrast to other models such as Gentner's Structure Mapping Engine (SME) (Falkenhainer, Forbus, & Gentner, 1990) that although on the surface seem to display human insights (such as drawing analogies between the solar system and the Rutherford atom), have only a superficial and syntactic understanding of the domains in which they function. Thus, although the domain of letter-strings may seem limited, Copycat has a deeper level of understanding of the concepts that are being manipulated than many other models of analogy-making.

A second important aspect of Copycat that sets it apart from other models of analogy-making such as SME or the Analogical Constraint Mapping Engine (Holyoak & Thagard, 1989) is that representation formation (how a situation is perceived) and mapping (what elements are equated across analogues) are modelled as co-dependent processes. This co-dependency is evident in human cognition (i.e. how we perceive DNA depends on whether we compare it to source code or a zipper), and reflects the context-dependent nature of perception (as discussed in chapter one).

Despite Copycat's success at modelling human performance on a range of problems, the approach suffers from two serious limitations. Firstly, like Jumbo and Numbo, Copycat uses a stochastic search with no memory of the avenues explored. Thus, when an impasse is reached, the system backtracks and tries again, often reaching the same dead-end. For example, in finding a solution to the problem *abc→abd, xyz→?,* Copycat's general biases force it to replace *z* by its successor in order to find a solution. However, in Copycat's Slipnet, the letter "z" does not have a successor, making the answer invalid. Copycat would deal with such an impasse by clamping the Temperature at a high value to release breaker Codelets that would dissolve the Workspace structures so that a new solution could be attempted. However, as Copycat had no memory of previous events, the system would often travel the same path many times before an alternative solution was found.

The second major limitation of Copycat lies in the way in which conceptual slippage is modelled. In this system, potential slippages are represented by explicit links in the Slipnet that can change length with the situation, making the slippage more or less likely to occur. The problem of this approach is that context is modelled at a global rather than local scale, proposing that the conceptual definition itself changes with context. This process does not generalise well to other examples of conceptual slippage where several instances of the same concept may be present under slightly different contexts. For example, in visual word recognition where context is local (where different instances of the same letter can occur in different positions), it does not make sense to globally modify the definition of a concept. This problem of relying on a global context has also been highlighted by French (1995) in the following real-world anecdote (1995, p43):

> "Suppose you are visiting a friend in Detroit, Michigan, and you wish to go to Ann Arbor. You ask for directions, and your friend replies, "Just go west on I-94." You happen to know that in the parking structure where you are parked, both the west exit and the east exit provide equally quick ways of getting out of the structure. But, even though the concept "west" is still active in your mind from your friend's directions, this does not affect your selection (or only minimally so) of exit from the parking structure. But how can this be, if there is only one "west" in the concept network of your brain? It would seem that if "west" were active, this should give a significant boost to the likelihood of choosing the west exit over the east exit. But this certainly does not seem to occur."

Thus, although the mechanisms employed by Copycat for conceptual slippage work in a range of letter-string analogy problems, they cannot be classified as a general account of context-dependent recognition.

2.3.4 Tabletop

Tabletop (French, 1995) is a direct descendant of Copycat which was aimed at testing the scalability of the FA approach to a less constrained environment. In this domain, two characters (Henry and Eliza) are sitting opposite each other at a table, with a collection of cutlery and dishes in front of them. The aim of the task is for Henry (the user) to point to an object on his side, and for Eliza (Tabletop) to find an object on her side that is equivalent. Thus, like Copycat, the task is one of analogy-making, but is less constrained in that the items can be placed anywhere in two dimensions rather than laying adjacent along a single dimension as in Copycat (see figure 2.7).

Tabletop identifies and tries to address several issues related to the scalability of Copycat. One such issue (mentioned previously) was the inability of Copycat to model local conceptual slippage. In the Tabletop domain, this issue becomes more relevant as there may be several instances of the same concept present, but in totally different contexts. For example, (as seen in figure 2.7), there are situations where locally, objects have diagonal symmetry with their analogues, but at a larger scale, are in mirror symmetry with other groups of objects. Thus at different levels of this

problem, there are different contexts, which should be represented by different activation levels of concepts in the Slipnet. However, as there exists only one instance of a node in the Slipnet, the conundrum occurs as to how these potentially "multiple activations" of each concept should be handled (French, 1995, p43-44).

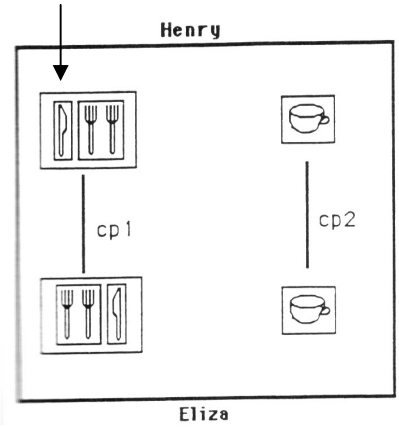


Figure 2.7. An example problem given to Tabletop (taken from French, 1995, p85). The aim of the system is to play the role of Eliza, and to point to an element analogous to the one pointed to by Henry. Like Copycat, this requires complex hierarchical groupings and correspondences to be perceived.

The approach taken in Tabletop to address the "multiple activations" dilemma, is to encode a form of a memory in which activations for a particular context are stored. When the program leaves one context (a level of the representational hierarchy), it is able to reconstruct the activation of the Slipnet concepts in the old context. In Tabletop, because of the small domain size, such a procedure is effective because it only relates to two Slipnet concepts: "mirror symmetry" and "diagonal symmetry."

A second important way in which Tabletop differs from Copycat is in its treatment of parallel investigations of problems. In Copycat, for a relationship to be perceived (such as a successor bond between two letters), all competing existing perceptions need to be destroyed. Tabletop differs from Copycat in the fact that any perception that has been destroyed is allowed to linger at a "subconscious level" for a certain period of time. This process was designed to reflect the fact that when viewing the old woman/young woman visual illusion, people are able to readily flip back and forth between the interpretations once each has been recognised. Such an ability was thought to suggest that "we maintain a somewhat active (although below-conscious-threshold) representation of the second figure in our brains" (French, 1995, p44).

The Success and Limitations of Tabletop

Tabletop differs from the previous FA models in that, instead of completely destroying competitors, "subconscious" copies of previously experienced perceptions are allowed to linger. This approach more closely resembles the gradual flow of information that is responsible for the contextual sensitivity exhibited by connectionist networks (discussed in detail in chapter three). For example, in speech recognition, it is beneficial to have several active competing interpretations at a phonemic level, so that possible interpretations at a word level can be partially activated. This partial activation can in turn, provide a context for disambiguating the phonemes themselves. If only one interpretation was allowed to occur at each level (as in models such as Copycat), it would be far more difficult to resolve such perceptual ambiguities.

Although it is clear that a "subconscious" search through multiple alternatives occurs in lower levels of processing (such as word recognition), how this approach generalises to higher-level tasks (such as planning) needs to be clarified. Firstly, in order to avoid a combinatorial explosion, it needs to be determined how many such subconscious interpretations are allowed to be explored in parallel and to how many hierarchical levels. Having this search be too broad is analogous to allowing the ant colony to scatter in all directions, rather than having the search and resources focused on a particular path. This issue is also relevant to the activations of the nodes in the Slipnet that are meant to reflect the current interpretation. If too many different interpretations are explored in parallel it is difficult to see how the search could be effectively regulated by a structure such as the Slipnet.

2.3.5 Metacat

One of the main limitations of the Copycat model is that it uses a stochastic search strategy with no memory of the avenues that it has explored. This leads to the situation that when an impasse is reached, the system backtracks and tries again, often reaching the same dead-end. The Metacat project (Marshall, 1999) was aimed at addressing this limitation of Copycat by adding self-watching into the model so that it could identify and bypass investigated avenues that were not fruitful.

Metacat implements a form of self-watching by extending Copycat's architecture with a further three components: the *Themespace*, the *Temporal Trace* and an Episodic Memory. As well as the typical Slipnet concepts, Metacat has a collection of themes that represent essential ideas underlying answers. For example, there is an *Alphabetic_position:opposite* theme that represents the fact that correspondences in the solution occur between objects that are at opposite ends of the strings (such as mapping *c* and *k* in the strings *abc* and *kji*). Over the course of a run, the Themes that are used in the construction of a solution are stored in a Workspace-like structure called the Themespace. These identified Themes can be used to trigger episodic memories stored from previous solutions. This allows Metacat to suppress solutions that are too similar, so that other solutions may be investigated.

Metacat's Temporal Trace is a short-term memory system (an extension to the Workspace) that encodes the temporal sequences of important processing events that have occurred during the formation of the solution so far. Such events include, for example, chunking a whole string of letters, generating a rule, or hitting a snag (trying out a rule that does not give an answer) (see figure 2.8).

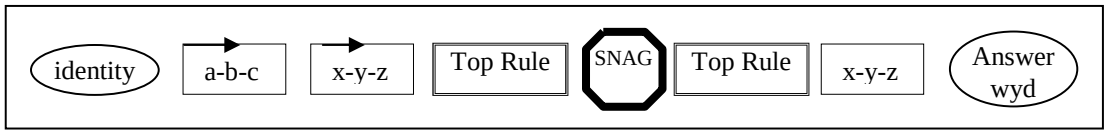


Figure 2.8. An example of the temporal record on the run of the problem abc→abd, xyz→? (Adapted from figure 1, Marshall, 1999). This shows that initially the letters in the same spatial positions are mapped (identity theme) with the two strings being perceived as right-going successor groups. A "top rule" (a rule for the initial transformation) is generated ("replace rightmost letter by successor") that leads to a snag (when a transformed rule is generated). Processing continues, with a new rule being generated ("replace rightmost letter with d") allowing a final solution to form.

During processing, the Temporal Trace can be used to identify and record what important events occurred during solution formation, storing an abstract description of the answer explored. This trace also allows Metacat to "see" what it is doing throughout the generation of a solution and to respond, if necessary, by clamping themes and concepts at high levels of activation to place top-down pressure on the system; an approach particularly important in navigating around salient dead-ends. In the situation where a snag recurs due to similar Themes, this similarity may be noted with the system then clamping the offending themes with negative activation. This

negative activation suppresses the processes that build the offending structures in the Workspace, allowing alternative answers to be explored.

The Success and Limitations of Metacat

The main strength of Metacat is the inclusion of an episodic memory that can be used to store the paths to solutions as well as regulating resources to explore uncharted territory. Apart from allowing the system to avoid previously found impasses, such a system is ideal for modelling planning and game play, in that it provides a mechanism for evaluating and remembering many different possibilities before a strategy or next move is executed.

Although including an episodic memory in the Fluid Analogy models is advantageous, it is not obvious how the solution employed by Metacat could generalise to other domains. In particular, Metacat relies on themes such as *Alphabetic_position:opposite* to suppress or facilitate the activities of Codelets. Relying on such higher-order descriptions may be limiting, as it is difficult to imagine what general themes would be relevant to supporting the formation of correspondences in real world domains (such as choosing the appropriate set of mappings to draw an analogy between the solar system and Rutherford atom). Furthermore, in domains such as Jumble and Numble, it is also unclear whether or not such higher-order descriptions exist, or whether the hierarchical representations formed in working memory are sufficient on their own to be saved as an episodic trace.

2.4 A Summary of Central Mechanisms

The Fluid Analogy models represent a collection of systems that provide mechanisms to account for five out of the six cognitive abilities required for intelligent behaviour described in chapter one (i.e. omitting mechanisms to account for the processing of raw distributed information). As a whole, the mechanisms underlying the five cognitive abilities captured by the approach can be summarised as follows:

The ability to manipulate complex hierarchical information

- Working memory contains a set of symbolic structures, representing objects, their properties and the relationships between them.
- Manipulation and construction of working memory structures occurs in parallel through the actions of a set of perceptual agents (Codelets).
- Codelets are similar to productions in that they bind to any location in working memory that satisfies a set of preconditions, attempting to modify or form symbolic structures as a result.

The ability to exhibit rational, goal-directed behaviour

- Goal-directed behaviour emerges through the parallel interaction of Codelets (i.e. there is no central executive).

The ability to select and attend to relevant aspects of the world

- The formation of many types of structures in working memory are mutually exclusive (e.g., a relationship in Copycat cannot be simultaneously perceived as a right-going successor bond and a left-going predecessor bond).
- In cases where incompatible structures can form, Codelets compete in probabilistic battles to determine which structures prevail.
- Due to local competitions that result in a single structure being formed, only a single avenue of exploration through the solution space is attended to at a time.
- Some actions are more likely than others (e.g., in Numbo, operations that create a number closer to the target will be more probable), biasing the search.

The ability to process information in a contextually sensitive manner

- Concepts are represented as nodes within a semantic network.

- Links between concepts can encode similarity (i.e. the probability of being equated across domains during analogy-making). Such links are referred to as "slip links" as they define potential conceptual slippages.
- Slip links can dynamically change in length depending on the context making mappings more or less likely to occur.
- Active concepts in the sematic network affect the rate and probability at which related Codelets function, biasing the formation of structures in a top-down manner.
- Codelets achieve their tasks in a number of interleaved steps. The decision of which Codelet to run next is probabilistic based on the calculated urgency value. This approach allows different structures to form at different rates, asserting different pressures on the formation of a solution.

The ability to perform self-watching and mental regulation

- The activity of agents is regulated by the activation of related concepts in the semantic network.
- The formation of important structures is stored in an episodic trace.
- If snags are found in the episodic trace (e.g., repeated exploration of dead-ends), the related concepts are inhibited, allowing the exploration of alternative solutions.

2.4.1 Noted Limitations of the Approach

Although the general approach taken by the FA models has been proven to work in a number of domains, there are a few limitations that need to be addressed. Firstly, like other UTCs, these models only process symbolic information providing no mechanisms for dealing with the raw distributed information provided by the senses. Secondly, as discussed, the mechanisms employed by Copycat, Metacat and Tabletop to handle conceptual slippage only capture global context, and do not generalise well to instances of low or high-level cognition that require such pressures to be local.

2.5 Summary

This chapter overviewed the main models of high-level perception developed by the Fluid Analogies Research Group. These models are relevant to the implementation of new cognitive modelling frameworks as they provide mechanisms to deal with important abilities crucial to intelligence that are not typically captured by UTCs (such as the ability to process information in a contextually sensitive manner and to perform self-watching and mental regulation). The main power of these models stems from their modelling of cognitive phenomena at a distributed level, through the parallel interactions of many agents. Such agents provide local support and competition for each-other, and create structures in a gradual and context sensitive manner. In this respect, the FARG models differ substantially from UTCs such as ACT-R and SOAR, and follow a more flexible connectionist-like approach.

Chapter 3
Computational Models of Cognition

The Fluid Analogy models are just one of many frameworks for cognitive modelling. The primary aim of this chapter is to bring into light a range of research that is relevant to addressing the main limitations of the Fluid Analogy approach in the development of a new computational framework. Such research provides inspiration for how to represent tasks and system knowledge in a domain independent manner, and how to process information that is context-dependent or distributed in nature. As many of the reviewed models can be considered competitors for the framework that is later proposed by this thesis, their strengths and weaknesses in capturing the six core cognitive abilities detailed in chapter one is also discussed.

The first section of this chapter is devoted to connectionism, highlighting the features that have proven useful in capturing context sensitivity and the processing of raw distributed information. The further sections then review general cognitive architectures such as SOAR and ACT-R that have proven successful in modeling aspects of high-level cognition, highlighting the data structures and scripting languages that afford the manipulation of complex symbol structures. The overriding features that are identified as being useful in supplementing the deficits of the Fluid Analogy approach are then summarised.

3.1 Connectionism

Although connectionism has not so far proven as effective as the symbolic approach at modelling such tasks as planning, problem-solving and reasoning, it does provide mechanisms for modelling context dependencies and the processing of raw distributed

information. The basic properties of the connectionist models that allow these important aspects of cognition to be captured are described below.

3.1.1 Mechanisms Accounting for Context Dependent Recognition

An early influential computational model investigating the effects of context on perception is the interactive activation model (IA) developed by McClelland and Rumelhart (1981). This model was used to simulate the contextual dependencies that occur in language processing. In particular, the IA model simulates the word superiority effect; the phenomenon that subjects are more accurate at recognizing letters that appear in words compared to letters in isolation. The IA model was developed in accordance with four basic assumptions about cognition that were believed by the authors to underlie context-dependent processing:

1. *Perception occurs in a multilevel processing system.* According to McClelland and Rumelhart, this tenet "is nearly ubiquitous" and uncontroversial, asserting that in domains such as language processing there are separate levels of representation for distinct entities such as letters, words, and higher order structures such as sentences.
2. *Deeper levels of processing are accessed via intermediate levels.* This assumption asserts that in processes such as reading and speech recognition there are multi-level representational hierarchies in which simple features are combined into more complex structures. For example, in reading, it is believed that words are not recognised directly from the visual input, but rather, are filtered through several different representational layers that extract features of increasing complexity (i.e. edges, lines and bars, letters etc.).
3. *Processing is interactive.* Rather than perception being a totally data-driven or feedforward flow of information, McClelland and Rumelhart assert that there is a bi-directional flow of information between layers, allowing for top-down influences upon perception. For example, the word superiority effect can be understood in terms of active word candidates placing top-down pressure to facilitate the processing of individual letters. This process is considered interactive as the word candidates become active from the active

letters themselves, thus resulting in a cyclical flow of information that accounts for the effect.

4. *Information flow is continuous*. In order for the previous tenet to hold true and for top-down information to have an effect on lower-level processing, information flow must be gradual and continuous. That is, for word candidates to have an influence on the processing of letters in the word superiority effect, word candidates must become active before the letters have been fully processed. That is, letters cannot become active in a single step, but rather need to gain activation gradually over time. Implicit in this assumption is the further requirement that although some transformations could be considered symbolic (such as the rules for recognizing individual words from a sequence of letters), partial matching is required in the spread of information between representations. Such partial matching is required so that partially active concepts (i.e. letters) can activate concepts at a higher representational level (i.e. words). This partial matching is also required in the feedback flow of information, so that words are not required to be fully processed before they have a top-down influence on the processing of the individual letters that they contain.

The IA Model of the Word Superiority Effect

The IA model utilised by McClelland and Rumelhart (1981) exemplifies the ease at which the above four properties can be captured within the connectionist framework. In modelling the word superiority effect, McClelland and Rumelhart identified three distinct layers of processing: the feature level, where the features of letters are detected; the letter level, where individual letters are recognised; and the word level, where individual words are detected. Reflecting these distinct levels, the model is divided into three separate layers, with nodes representing individual features, letters and words. The model is capable of recognizing four letter words, with separate nodes representing visual features and letters at each of the four positions. The layers are connected in a feed forward and feed backward manner through weighted links that allow for an interactive flow of information. There are also inhibitory connections within and between the letter and word levels to implement competition

between mutually exclusive interpretations (see figure 3.1). In the network, there are no direct links between visual features and words, requiring that the deeper levels of processing be accessed by intermediate levels.

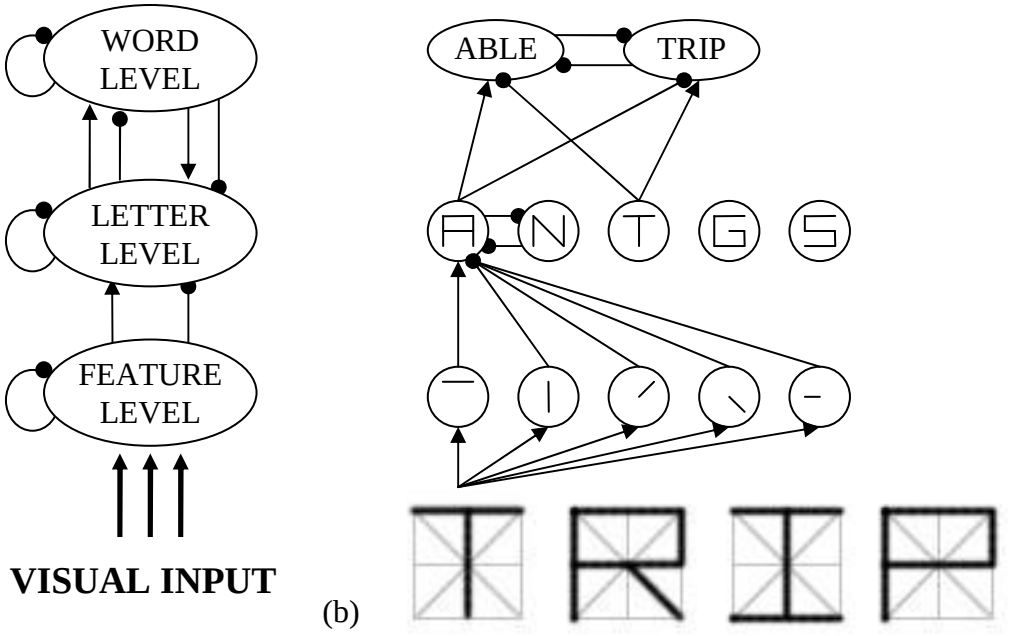


Figure 3.1. The basic architecture of McClelland and Rumelhart's interactive activation model of the word superiority effect (adapted from McClelland & Rumelhart, 1981). (a) Apart from a feedforward flow of activation between layers, there is a top-down flow of information between the word and letter level, and inhibitory connections within layers to implement competition between mutually exclusive interpretations. (b) In the network there are separate nodes for each of the letter features and letters in each of the four letter positions (not all nodes and links shown).

The IA model is implemented as an interactive activation and competition network (IAC network), which allows nodes to take a graded level of activation, and for information to flow between the nodes in a continuous manner. Each node has a varying level of activity that is influenced by the net input of activation received from connected nodes. The net input to each of the units at a given time, $n_i(t)$, is given by the equation

$$n_i(t) = \sum_j \alpha_{ij} e_j(t) - \sum_k \gamma_{ik} i_k(t) \qquad (1)$$

where $e_j(t)$ is the activation of an excitatory node, $i_k(t)$ is the activation of an inhibitory node, and α_{ij} and γ_{ik} are the associated weights. When the net input is

positive, the internal activation will climb towards its maximum allowable value (at a rate determined by the net input), whereas if it is negative, it will decay over time.

In processing information in the IA model, activation flows from the active letter features, exciting the letters that contain them. As there is a degree of overlap between the visual features of letters, initially several letter nodes may become partially active for each letter position. This partial activation, in turn, flows to the word level, activating likely word candidates. Activation also is capable of flowing backwards from the word level, reinforcing the activation of consistent letters. Thus, if there are letters that are ambiguous based on the active features, activation from the word level can provide an extra source of information for their discrimination. Due to inhibitory connections between letter and word interpretations, as the strongest match gains activation, it inhibits the other alternatives to a greater degree, gradually settling into a single decision at each layer.

In modelling the context-dependent nature of cognition, the IA model still remains one of the most influential accounts, with most current approaches being consistent with the four assumptions mentioned above. For example, the processes of semantic priming (the phenomenon that a word is processed more rapidly when temporally preceded by a semantically related word compared to a semantically unrelated item), has been successfully modelled utilising attractor neural network models (e.g., Cree, McRae & McNorgan, 1999; Plaut 1995). Such models, although utilising more sophisticated representations and training procedures, still adhere to the four basic assumptions mentioned above, relying on a continuous and interactive flow of information. Also consistent with the four assumptions, is the view of O'Reilly & Munakata (2000, p14-18), who cite *parallelism*, *gradedness*, *competition* and *interactivity*, as being the crucial general aspects of cognition that give rise to such context dependencies.

3.1.2 Processing Raw Distributed Information

In utilizing the same kind of mechanisms employed by the IA model (e.g., parallelism, gradedness, competition and interactivity), connectionism can also naturally account for the processing of raw distributed information. In such cases,

however, (particularly in large networks or when the appropriate patterns for activation in the intermediate layers is not known a priori) learning algorithms that set appropriate connection weights are required. Typical learning algorithms include Hebbian Learning (Hebb, 1949) and Backpropagation (Rumelhart, Hinton & Williams, 1986).

One example connectionist approach that naturally processes distributed data is Leabra (**l**ocal, **e**rror-driven and **a**ssociative, **b**iologically **r**ealistic **a**lgorithm) developed by O'Reilly and Munakata (2000). This model is important in that it models a wide range of cognitive phenomena utilizing the same activation update and learning algorithms. Such phenomena include the visual processing of natural scenes learning encodings that are similar to those found in the human visual cortex, the extraction of distributed semantic representations of words from natural text, and the modeling of aspects of episodic and semantic memory (O'Reilly & Munakata, 2000). Like in the IA model, the processing of information (which is distributed in this case) is achieved through the parallel activity of multiple units whose internal activation levels are updated slowly over time based on their net inputs. This model differs from the IA model however, in that it attempts to model the biological level more closely, with inhibition being regulated within each layer (rather than allowing inhibitory connections between layers), and with the updating of the internal activation of each unit reflecting the electrophysiology of the neuron.

Due to their role in accounting for both the processing of raw distributed information and for modeling contextual sensitivities, the properties of parallelism, gradeness, competition and interactivity are evidently useful features to incorporate into future frameworks that attempt to model such phenomena.

3.2 Symbolic Models of Cognition

In modeling human high-level cognition, very few models take a purely symbolic approach due to its inherent inflexibility (as discussed in chapter one). Such models that do take this approach include SOAR (Laird, Rosenbloom & Newell, 1986; Laird, Newell & Rosenbloom, 1987; Newell, 1990) and EPIC (Meyer & Kieras, 1997).

These models differ from each other in that SOAR was developed to perform high-level problem-solving, whereas EPIC was designed to account for human reaction times on perceptual-motor tasks (and as a result incorporates a number of perceptual and motor modules). The following sections focus on SOAR, as it provides a greater explanation (from the symbolic perspective) of the processes underlying mental deliberation, with EPIC using a simplistic symbolic processor that is limited in its problem-solving abilities.

3.2.1 SOAR

Soar is a generic architecture for intelligent behavior that was first implemented as a computer program in 1983 (Laird, Rosenbloom, & Newell, 1986). The architecture is based on two fundamental assumptions: that humans can approximately be described as *knowledge level systems*, and that they process information symbolically (Newell, 1990, p113-117). The first of these assumptions is that humans use their knowledge in a rational manner to achieve their goals, desires and intentions. In many cases however, such goals cannot be attained in a single step, instead requiring a hierarchy of nested subgoals in order to direct immediate behaviour. For example, the task of catching a bus to work can be broken down into a sequence of several high-order actions (such as walking to the bus stop, and purchasing a ticket), that can each be further segmented into more immediate actions (e.g., climbing stairs or getting money out of a wallet). Due to the centrality of such task decompositions in intelligent behavior, much of Soar's architecture has been specifically designed to allow the system to plan and navigate through a path of hierarchical subgoals in order to process higher-order tasks.

The second major assumption that governed the design of Soar was the belief that human cognitive processing could be described effectively as symbolic. According to Newell (1990, p121-139), processing in the brain can be described at different levels (or layers) of abstraction, with the most fundamental mental deliberations occurring at the cognitive band (which lies directly above the biological or neural band). Newell suggests that such deliberations take no less than 100ms to perform and are concerned with bringing available knowledge to bear on situations in order to choose certain

actions over others. Due to this time scale, Newell also postulated that such deliberations occur in parallel, as many more than 100 serial processing steps are required to perform such complex tasks as generating the next move in a game of chess (which takes roughly 10s to perform). In accordance with these beliefs about human cognition, SOAR processes its information symbolically, using parallel operations that emulate the fundamental deliberations upon which cognition is supposedly based.

The architecture of SOAR

According to the symbolic view of cognition, intelligent deliberation can be described as the selection and application of discrete operators on a problem state (such as determining the next move in a game of chess). In SOAR, such states include a symbolic description of the perceived environment, as well as the internal state of the agent (including its current goals and mental perceptions about the situation). Operators can modify the current state, either by sending signals to the motor system, or by directly manipulating the mental representations in working memory.

In formulating a solution, SOAR utilises a set of architectural components and processes that are independent of the cognitive task being performed. In particular, SOAR contains a single storage of long-term memory (encoded in terms of productions), a single representation of working memory (that symbolically describes the current situation in terms of objects, their attributes and interrelations), a single mechanism for generating goals (*automatic subgoaling*), and a single learning process (*chunking*). What differs between the tasks that SOAR can perform, is the knowledge that is represented in long-term and working memory.

Working Memory

SOAR's working memory stores information about the current situation being perceived. Such information is represented in terms of *working memory elements* (*WMEs*) that denote the properties and relationships between perceived objects. WMEs are encoded as symbolic *identifier-attribute-value* triples, representing the

properties of an object (such as the fact that *B1 is_a Block*, or *B1 is_coloured Red*) and the noted relationships between them (such as *B1 is on_top_of B2*) (see figure 3.2). Such structures are useful in that they are capable of representing the components of a wide selection of problem solving tasks, ranging from games (e.g. chess) to mathematical puzzles and robot control tasks. This domain-independence in representation is an important feature of the model.

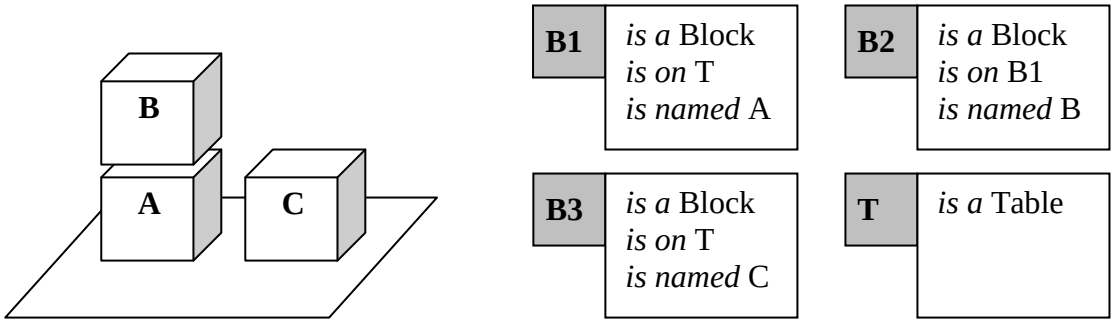


Figure 3.2: An example representation of a blocks world situation in SOAR. (a) displays a visual depiction of the situation. (b) illustrates the symbolic representations used by SOAR. Each individual object has a unique identifier (such as *B2*), and a set of attributes that represent properties of the object and relationships to other objects.

Operators

During the processing of a problem, SOAR moves from one problem state to another through the application of operators. Operators represent tasks that can be performed by the system, such as the physical action of moving one block onto another, or the mental prediction of the effects of such an event. In order to implement goal-directed behaviour, the selection of operators is guided by "the principle of rationality" (Newell, 1990, p33), which favours the application of operators that are likely to lead the system closer the obtainment of its goals.

Operator Selection and the Decision Cycle

One important feature of SOAR is that it asserts that the application of operators is a serial process, with only one operator being able to be applied to the current state at a

time. In selecting the next operator, SOAR uses an architectural feature called the *decision cycle* that is divided into two main phases: *elaboration* and *decision*. During elaboration, all relevant information from long-term memory that is useful in determining what operators are available and which ones are preferred over the others is brought into working memory. Such information is brought to bare on the situation through the use of symbolic if-then rules, called *productions*. This process is iterative, with all productions stored in long-term memory that have their conditions satisfied being fired in parallel, with the associated changes to working memory being applied. Such productions can take one of four forms. Firstly, productions may add relevant WMEs to working memory that may be important in the selection of an operator (such as the inference that "if it is night, it will be dark outside"). These pieces of information are generally temporary, being removed on subsequent cycles if their conditions are no longer matched. The second form of production is capable of proposing valid operators and adding a description of them into working memory (e.g., describing all possible valid next moves in a game of chess). These proposed operators are compared by the third form of production, capable of specifying preferences for one action over another (such as preferring the taking of the King over a Pawn).

At the end of the elaboration cycle (when no further productions can fire), a decision is made if there exists a single operator that is preferred over all others, with the action for the selected operator being applied. As with all knowledge in SOAR, operator actions are also specified in terms of productions.

Impasses and Subgoaling

At the end of the decision cycle, occasionally the preferences in working memory do not yield a single operator. Such impasses occur when several operators are proposed, but there is not enough information to select between them. SOAR deals with such situations by creating a subgoal that aims at resolving the impasse. Subgoals are implemented in terms of a nested hierarchy of states that explore the impasse condition in detail. That is, when an impasse occurs, a totally new state is created, holding the WMEs that describe the problem to be resolved. For example, if working

in a blocks world domain (as in figure 3.3 below), there may be several potential actions that can be made. If it is not obvious which action to take a new state may be created that notes these possibilities. Generally, the aim of such impasse states is to create preferences that decide between the operators. In the blocks world case, this may be achieved by conducting a look-ahead search, and evaluating which operator takes the current state closer to the overall goal. Such a search is conducted by serially exploring the result of each operator. Apart from a look-ahead search, there are several other ways of resolving impasses, such as randomly selecting between operators, or drawing in information as to the success of the operator in similar conditions. All such methods of resolving impasses are also coded in terms of productions.

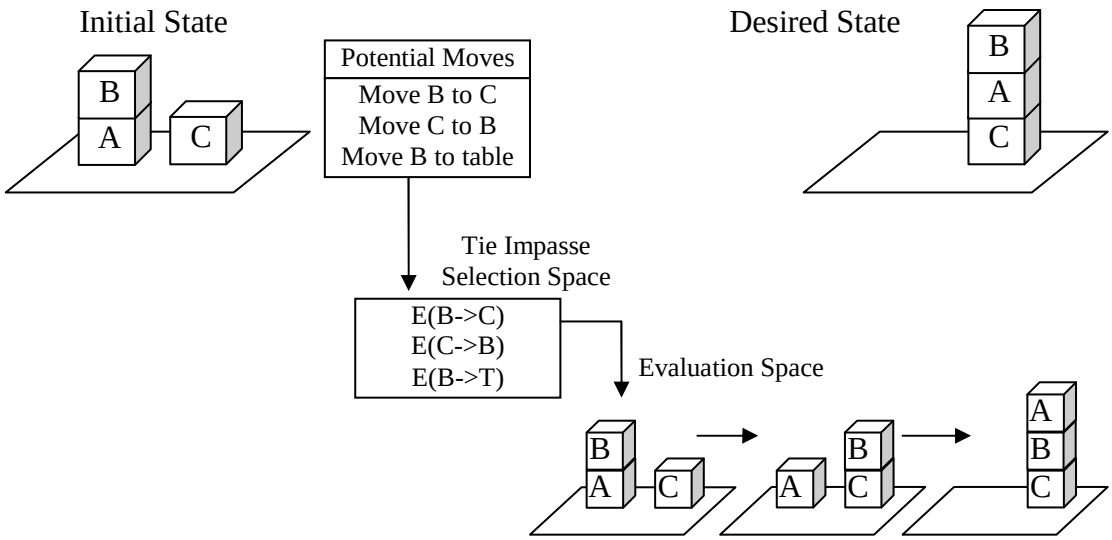


Figure 3.3. Subgoaling in SOAR. In this example, three moves are possible. If there is no information as to which operator is preferred, a "Tie Impasse" is detected with a new state being created, with the goal to evaluate each move to determine the best choice. In this state the operator E (evaluate) is proposed for each of the three potential moves, with the operators being tested sequentially. In the first case, B->C is attempted for the next two moves (until no further moves can be conducted that do not involve backtracking). As this does not lead to the desired state, this operator is removed from the original choices, and with the next choice likewise being tested.

The Success of SOAR

The architecture of SOAR affords the manipulation of complex symbol structures and the heuristic search of discrete state spaces in which a single operator is needed to be

selected from. For this reason, SOAR has had much success in modeling problem-solving tasks such as the blocks world, the Tower of Hanoi, Tic Tac Toe, missionaries and cannibals, and eight queens (Kaird et al., 1987). Much of SOAR's power in manipulating complex symbol structures lies in the use of productions to encode the required transformations. In the simplest form, the conditions and actions of a production relate to the presence or absence of specific objects or object properties in working memory. For example, in the blocks world domain, if blocks A and B are both clear (i.e. have no other block on top of them), the operation "move A to B" might be proposed. Such rules however, can be made far more general with the inclusion of variables that can bind to objects or attribute values. For example, the rule "if <o1> and <o2> are clear, propose move <o1> to <o2>" will result in a move operator being proposed for **any** pair of free blocks. Compared to connectionist networks in which rules are specified locally through weighted connections, variable binding provides a powerful mechanism for manipulating complex symbol structures, as patterns can be detected invariant of position. Furthermore, due to their symbolic nature, productions are often fairly intuitive and transparent, and can be readily coded in their applications to different domains. The format of a production involving variables can be seen in figure 3.4, in the coding of a rule that performs counting.

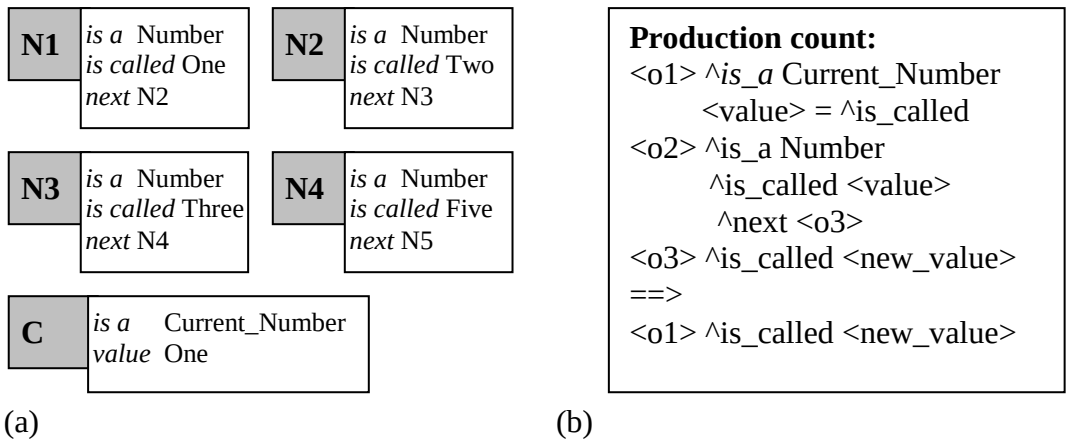


Figure 3.4. Variable binding in a counting task. (a) represents the contents of working memory required to perform counting: a sequence of numbers (N1 to N4) and the current number (C). (b) provides a single production capable of performing the task of counting using variable binding. <o1>, <o2> and <o3> are variables that will be bound to different objects in working memory, with <value> being bound to the value of the current number.

The Limitations of SOAR

The symbolic approach to cognition, exemplified by SOAR, attempts to model the "conscious" aspects of thought, which through introspection appear symbolic and serial in nature. According to O'Reilly and Munakata (2000, p14) this approach is ultimately flawed, as it is attempting to model "the proverbial tip of the iceberg" while ignoring the "great mass of cognition … that floats below… [that] is necessary to keep the tip above water in the first place". According to O'Reilly and Munakata, such underlying aspects of cognition include parallelism, gradedness, interactivity and competition. As stated earlier, these are the very features that are required to model contextual sensitivity and the processing of raw distributed information; two of the abilities inherently lacking in the symbolic approach.

3.3 Hybrid Models of Cognition

Over the years there have been many attempts made to merge the flexibility of connectionism with the representational power of the symbolic approach. Such models include 3CAPS and 4CAPS (Just, Carpenter & Varma, 1999; Just & Carpenter, 1992), DCPS (Hinton & Touretzky, 1987) and BRAINN (Bogacz & Giraud-Carrier, 1998). This section focuses of two such hybrid models of cognition that have gained prominence for their success at modeling a wide range of cognitive phenomena; ACT-R (Anderson, 1993; Anderson & Lebiere, 1998) and DUAL (Kokinov, 1994b).

3.3.1 ACT-R

Anderson's ACT-R model (Anderson, 1993; Anderson & Lebiere, 1998) has common roots with SOAR, with both incorporating a production based account of procedural memory. ACT differs form SOAR however, in that it architecturally differentiates between procedural and declarative memory. Declarative memory in this model is implemented in terms of a semantic, spreading activation network. This network is important in that it allows a form of subsymbolic processing to occur, with the accessibility of information being statistically graded and context sensitive.

The Architecture of ACT-R

SOAR and ACT-R are similar in that they model cognition as resulting from the interaction of procedural knowledge, with instances of declarative knowledge. How these systems differ is where and how the chunks of declarative knowledge are stored and accessed. In ACT-R, declarative knowledge is stored in a set of independent and encapsulated modules that are devoted to such functions as object recognition, motor output, retrieving information from memory, and keeping track of the current goals. The role of the production system is to tie these independent modules together, modifying their contents based on the contents of the other modules. However, not all of the information from these modules is available to the production system at any given moment. Instead, each module has an associated buffer that holds the information the system is attending to at the time, which is generally only a single chunk of information. This restriction reflects the fact that humans have limited awareness, for example, being conscious only of the information currently retrieved from long-term memory as opposed to all stored information. ACT-R does not specify how many such buffers exist, but has a number of different modules that have been developed as part of the standard implementation. Such modules include the goal buffer which acts as a working memory (holding the internal state of the system when working through a problem), the retrieval buffer (which holds the chunk of information that has last been retrieved from memory), two visual buffers (which represent the outputs of the "what" and "where" visual pathways), and a manual buffer (representing motor commands).

The Goal Buffer

ACT-R exhibits goal directed behaviour utilizing a "Goal Buffer" that stores the current objectives of the system. However, as many goals cannot be achieved in terms of a single step, ACT-R's goal buffer stores a goal hierarchy, holding the current goal as well as a stack of higher-order goals that are required to be completed. The Goal Buffer acts as a form of working memory, holding the information that is currently being mentally manipulated. For example in adding two numbers together, the Goal buffer may hold a chunk with three slots (implemented as a single node with

three links): two of which store the numbers to be added, with the third storing the result (see figure 3.5). Unlike the other buffers in ACT-R's architecture, the Goal Buffer may also store more than a single chunk of information, allowing other chunks to be contained as sub-elements. This flexibility allows complex hierarchies of structure to form, such as those required in sentence processing tasks.

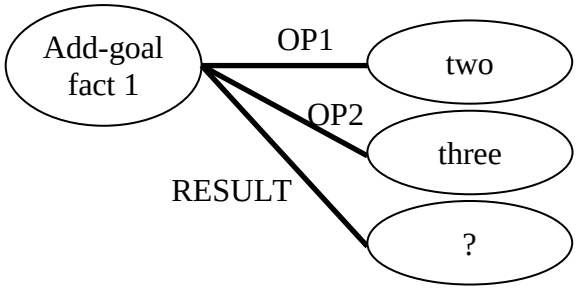


Figure 3.5 ACT-R's Goal Buffer. A goal is defined by a single concept node, and the nodes linked to its various "slots".

The Declarative Memory Module

In ACT-R, declarative memory is stored symbolically in a network of interconnected nodes. Each learned fact is represented by a separate node in the network that is linked to conceptually related elements. That is, each fact can be thought of as a "chunk" that has a set of linked concepts that denote its type, its properties and relationships to other chunks (see figure 3.6).

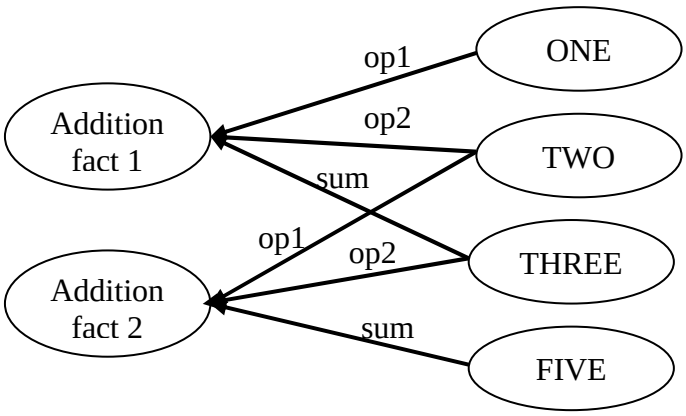


Figure 3.6 Declarative memory in ACT-R. Declarative memory is stored in terms of a network of interconnected nodes. The network above stores the facts that "1+2=3" and "2+3=5"

In ACT-R each node has a level of activity reflecting the accessibility of the information to memory retrievals. The activation of each node (A_i) is influenced by two main factors; the base activation of the node (B_i), as well as activation that has spread from concepts active in the goal buffer (a product of their attentional weightings in the goal buffer, W_j, and the strength of the corresponding association, S_{ji}):

$$A_i = B_i + \sum_j W_j \cdot S_{ji} \qquad (2)$$

The current chunk in ACT-R's goal buffer receives an attentional activation W, which is spread evenly between the chunks in its defined slots (see figure 3.7). Thus, if the current goal is to add the numbers "two" and "three", the nodes representing "two" and "three" in declarative memory will receive a portion of the goal's activation (W_j). Any node in declarative memory that is directly connected to a chunk referred to by the current goal will receive a degree of activation (the attentional activation W_j multiplied by the strength of the link between them). Thus, chunks which share several features with the current goal will gain a higher level of activity, and become more accessible during retrieval.

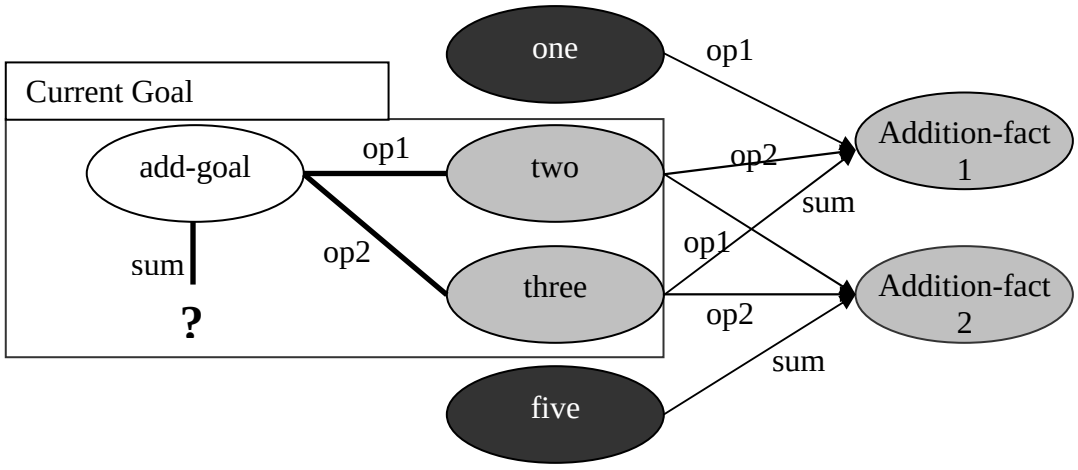


Figure 3.7. The spreading of attentional activation in declarative memory used in adding the numbers *two* and *three*. Firstly, the attentional activation of the current goal node is spread evenly between its connected features (*two* and *three*). This activation then spreads to connected nodes in declarative memory, raising the accessibility of relevant chunks.

In retrieving information from declarative memory, a retrieval request is made by the production system, such as to return an "addition fact" whose "op1" value is "two", and whose "op2" value is "three" (i.e. in order to return the fact corresponding to the addition of these numbers). Such retrieval requests are met by the return of a single matching node, emulating the apparent serial nature of retrievals from human memory (e.g., the serial recall exhibited when trying to remember the names of previous presidents of the USA). In selecting a node to return from the set of possible matching alternatives, only the nodes matching the actual request are considered. For example, in figure 3.7 above, Addition-fact 1 (1+2=3) and Addition-fact 2 (2+3=5) are equally active, due to the spreading activation from the numbers "two" and "three" in the current goal. However, only Addition-fact 2 would be returned by the retrieval, as it is only this fact that has an "op1" of "two" and an "op2" of "three".

When several nodes match the retrieval request, the decision of which node to return is determined probabilistically based on their activation levels. Specifically, this selection process is achieved by adding Gaussian noise to the activation of each node, and returning the node with the highest resulting value. As the elements of retrieval requests are generally the elements contained in the current goal (e.g., "two" and "three"), nodes matching the request will have high levels of spread activation, resulting in a high probability of being selected.

Procedural Memory

As stated, ACT-R is similar to SOAR in that it has a procedural memory that is implemented in terms of productions (if-then rules) that are used to modify the declarative information. In ACT-R, productions are used to test the various buffers for patterns, and act by either modifying the chunks in the buffers directly, or trying to retrieve specific chunks from declarative memory. Unlike SOAR in which productions are fired in parallel with the aim of resolving a single operator, ACT-R selects only a single production to fire at a time. As a result, ACT-R is limited to modelling fairly simple cognitive tasks (such as emulating serial recall tasks).

In order to select from the set of productions that may match the current situation, a form of conflict resolution is implemented in which the production with the highest *utility* is selected. The utility is a noisy, continuous value assigned to each production, that plays a similar role in production selection as activation does in chunk retrieval. Such utility values can be learned with experience, taking into account the expected cost in achieving the current goal (measured in time) as well as the likelihood of success (the calculated probability that the firing of the production leads to the desired objective). At each cycle of production selection, Gaussian noise is added to the utility values, so that although the production with the highest resulting value is always chosen, the actual production chosen can vary.

The Strengths and Weaknesses of ACT-R

In ACT-R there are empirically derived functions for calculating the time taken to perform the actions of productions and retrievals from memory. This prediction of reaction times has made ACT-R a fairly powerful tool in the area of human computer interaction, where models are constructed to evaluate expected human performance at complex tasks such as flying commercial aircraft or in aiding with the design of effective computer interfaces. ACT-R can also be used for the construction of cognitive agents, which are useful for group training exercises, and in multi-agent video games. Apart from in implementing such agents, ACT-R is capable of capturing a range of cognitive phenomena from visual search (Anderson, Matessa & Lebiere, 1997) to memory retrieval tasks such as the fan effect (Pirolli & Anderson, 1985), list memory experiments (Anderson, Bothell, Lebiere & Matessa, 1998), and implicit memory experiments (Lebiere & Wallach, 2001). Category learning (Anderson & Betz, 2002) and sentence processing experiments (Anderson, Budiu & Reder, 2001) have also been simulated.

Although ACT-R has been successful in emulating human reaction times on a number of tasks, several limitations of the approach are apparent. Firstly, unlike SOAR, ACT-R has not been designed to manipulate the hierarchical structures required for complex problem solving, limiting the approach to tasks requiring fairly simple representations. Secondly, like the Fluid Analogy models, context effects are derived from the varying activation levels in the semantic network. Although such an

approach is adequate for problems requiring global context, it does not provide the necessary mechanisms to account for local context effects (such as in the word superiority effect). ACT-R is also limited in the fact that all the representations are symbolic (and hence, not permitting the processing of raw distributed data), and that it does not have any explicit mechanisms for self-watching and mental regulation (which is required for the intelligent search of problem spaces).

3.3.2 DUAL

Unlike SOAR and ACT-R, DUAL is a general cognitive architecture that, similar to the Fluid Analogy approach, views high-level cognition as emerging from the parallel interaction of numerous agents (Kokinov, 1994b; Petrov & Kokinov, 1999). The general approach to problem solving was designed to explore the tradeoff between flexibility (keeping the search as large and as open-ended as possible) and efficiency (restricting the number of candidate-solutions considered). In dealing with this tradeoff, as with the Fluid Analogy Models, DUAL integrates the following ideas into its architecture (Petrov & Kokinov, 1999):

(i) Multiple candidate solutions can be explored in parallel.
(ii) An approximated "promise" of each candidate solution is calculated on-line.
(iii) The computational resources are distributed unevenly among the candidate solutions, in favour of the more promising ones.
(iv) Promise estimates change over time with new information, re-allocating the computational resources.

Unlike the other hybrid models of cognition mentioned in this thesis, DUAL does not have a modular architecture, instead being composed entirely out of agents. The agents themselves are hybrid in nature, containing both connectionist and symbolic aspects, and are used for both representing and processing information.

At the connectionist level, DUAL agents can be viewed as nodes within a semantic network representing such entities as object classes, object instances, properties and relations. Such nodes are connected to each other through labeled links (see figure

3.8). Thus, at the representational level, DUAL is capable of representing the same set of problems as SOAR, ACT-R and the FA models. DUAL differs from the symbolic approach however, in that each node has a level of activation that is spread to its neighbours. Activation levels in the network are dampened through decay, with only nodes directly associated with the current goal receiving direct input, resulting in only a portion of the network gaining threshold activation levels at any given time. The active portion of the network is referred to as DUAL's *working memory*.

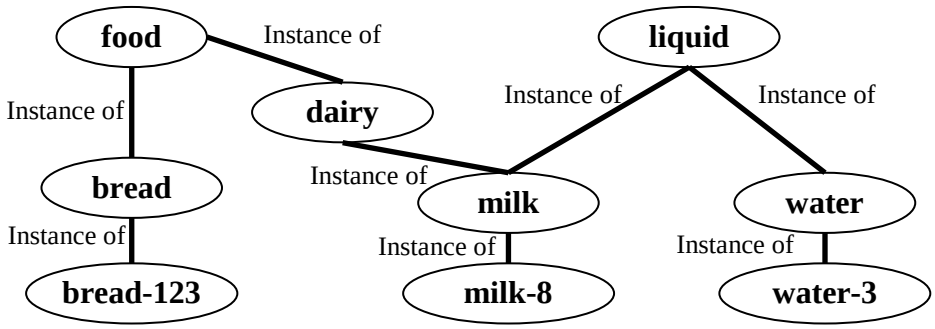


Figure 3.8. An example conceptual network utilized by DUAL. Nodes can represent categories (e.g., bread) or instances of the category (milk-8).

Unlike typical spreading activation networks, nodes in DUAL also act as agents that are capable of symbolic computation. Agents can deterministically construct, store, and transform symbol structures and exchange them with their neighbours, as well as creating or destroying existing links and creating other temporary agents that are integrated within the existing network. As with the Fluid Analogy models, such computations occur in parallel and at variable speeds. In DUAL, the speed at which an action can be performed by an agent is dependent upon the agent's activation level itself, with highly active agents performing their transformations at a faster rate. The actions of agents are programmed in S-LISP (suspendable LISP), an extension of common LISP, that suspends the symbolic computation for a specified duration. Thus, in implementation, the symbolic computations are instantaneously updated, but after a delay determined by the activation of the node, emulating the time taken to gradually perform the operation.

One example implementation of the DUAL architecture is in a model of analogy-making called AMBR (Kokonov, 1994; Petrov, 1998). This model solves problems such as generating a solution to heating milk given the scenario "there is some milk in

the teapot" and "there is a hot plate." AMBR solves such problems by drawing on information from other scenarios (such as boiling water in the teapot), firstly retrieving the appropriate source analog, mapping it to the target, and then transferring information to the solution.

In AMBR, a central part of solution formation corresponds to the process of finding correspondences between situations. This process is achieved by activating the nodes from the current situation, and allowing activation to spread to other scenarios. During this process, whenever a category instance becomes active, a symbolic marker is passed to its parent category, which is further sent up the hierarchy. Markers are also sent by category instances from the current situation. When two markers intersect at a category node, the origin of the markers is determined, with a new correspondence agent being formed, connecting the two category instances. For example, in figure 3.7 above, if *milk-8* is in the current scenario, correspondences are likely to form with *bread-123* and *water-3*. However, the rate at sending such markers is dependent on the activation of the agents, with *water-3* likely to have a higher activation level, and thus having its correspondence noted more quickly. In AMBR, correspondence agents are linked to each other, with consistent mappings through excitatory links and inconsistent mappings through inhibitory connections. This "parallel constraint satisfaction network" settles into a pattern of activation relating to a consistent set of mappings from which information can be transferred.

The Success and Limitations of DUAL

DUAL has been successfully used to model similarity judgments (Kokinov, 1992), analogy-making (Kokinov, 1994a) and deductive reasoning (Kokinov, 1994b). Due to its ability to model an interactive and continuous flow of information between distributed agents, DUAL, like connectionism, well-models context effects at both a global and local level. For example, AMBR (Kokinov, 1994a) is capable of creating analogical mappings by dynamically creating a parallel constraint satisfaction network. Such a network has the same properties as the IA model of McClelland and Rumelhart (1981) that was used to account for the word superiority effect. The AMBR model has also been used to display global priming effects in analogical

reasoning, with residual activation from previous events affecting the probability of their retrieval during problem solving (Kokinov, 1994a).

Despite DUAL's superiority over the Fluid Analogies models with respect to modeling local context effects, the architecture is limited in its ability to perform general problem solving. Unlike the FA models, DUAL is not able to perform self-watching or any form of backtracking, making it difficult for the model to search through discrete problem states. This limitation also prevents the architecture from performing analogy-making tasks, such as that performed by Copycat, in which explicit representations of the problem are formed dynamically. Instead, AMBR has been limited to mapping predefined scenarios in which the representations have been hand-coded. DUAL also does not readily afford the processing of raw distributed information, as although it is conceivable that agents could represent subsymbolic features, there are no learning algorithms for determining the appropriate weights between such agents as there are for complex connectionist networks.

3.4 Summary

This chapter was aimed at providing an overview of the main models that have been used for capturing intelligent deliberation. Although no single approach is able to perform all six of the cognitive abilities described in chapter one, as a whole these models do provide inspiration for specifying mechanisms that supplement the limitations of the Fluid Analogy approach. Firstly, in dealing with raw distributed information and context sensitivity, the properties of parallelism, gradedness, competition and interactivity have proven useful in both connectionist models and DUAL. As demonstrated by both the Fluid Analogy models and DUAL, the parallel interaction of agents has also been effective in modeling high-level cognitive tasks such as planning, problem-solving, reasoning and analogy-making.

A second conclusion of this review is that in representing high-level tasks in a domain independent manner, representing problems symbolically in terms of objects, their properties and relations has proven widely successful, with all hybrid system utilizing such an approach. In manipulating such symbol structures, it is evident that productions provide a powerful tool, allowing variable binding for object invariant

transformations using transparent rules that can be readily programmed and applied to new domains.

Chapter 4
An Overview of FAE

In creating a new cognitive modelling framework for capturing both high and low levels of processing, the Fluid Analogies Engine (FAE) is proposed. This framework includes mechanisms for modelling all six of the core cognitive abilities underlying intelligent behaviour that were specified in chapter one: the ability to process raw distributed information, the ability to select and attend to relevant aspects of the world, the ability to process information in a contextually sensitive manner, the ability to create and manipulate complex hierarchical representations, the ability to exhibit goal-directed behaviour, and the ability to perform self-watching and mental regulation.

The main architecture of FAE was inspired by the Fluid Analogy models, incorporating a working memory, semantic memory and episodic memory, with all high-level behaviour of the system emerging through the parallel interactions of local agents. The development of FAE was also inspired by connectionism and prominent UTCs, incorporating mechanisms that allow for the processing of raw distributed information and the capturing of subtle influences of context upon perception. FAE differs from typical hybrid models of cognition in that although information is described in terms of objects, properties and relations, such elements are internally represented as a distributed vectors of features. Thus, as well as representing the structures required for high-level processing, FAE can represent the raw distributed information required for low-level tasks. In FAE, such representations are manipulated through the use of a connectionist production system that dynamically constructs a neural network to perform the required transformations at a subsymbolic level. The production system also allows for variable binding and the creation and deletion of objects, properties and relations, affording the manipulation of complex

relational structures. As with Metacat (the most developed of the Fluid Analogy models), the production system is embedded within a general cognitive architecture including a semantic memory (that enables relevant information to be retrieved and utilised during problem solving), and an episodic memory (that allows the system to self-watch and backtrack out of dead ends).

This chapter provides an overview of FAE, detailing the constraints on its design, and the resulting architecture and algorithms that are employed.

4.1 Design Constraints on a Cognitive Architecture Supporting Flexible High-level Perception

Although the general architecture of FAE was inspired mainly by the Fluid Analogy models and current unified theories of cognition, it was designed specifically to support the six core cognitive abilities decribed in chapter one. As previous research has successfully explored differing subsets of these abilities, standard partial solutions in the form of architectural components and processes already exist (with those relevant to this project being mentioned in the previous chapter). It is the aim of the current project to identify the computational properties that have been successfully utilised in modelling each of the six mentioned abilities, and to unify them into a coherent modelling framework. The six cognitive abilities and the properties that underlie models that have successfully displayed them are detailed below. These properties can be viewed as constraints on the design of FAE, as many of the mentioned components or processes are non-orthogonal, mutually constraining the way in which each should be implemented.

4.1.1 The ability to process raw distributed information

The first ability considered in the design of FAE arose from the consideration that much intelligent behaviour arises at the subsymbolic rather than symbolic level, requiring representations and transformations that support raw distributed information (see Chapter one for the full discussion). According to Brooks (1990), such transformations are required for processing the information provided by the senses,

producing subtle motor commands, and for encoding the direct "automatic" pathways between these processes.

Out of the various modelling approaches discussed in this thesis, connectionism has proven the most successful at capturing low level cognitive behaviour (for reasons discussed in chapter three). In most connectionist models, units appear in discrete layers that allow complex transformations to occur through a sequence of simpler steps (e.g., Rumelhart et al. 1986; Lampinen & Oja, 1992). In processing sensory information, such layers are often topographic, supporting the detection and integration of local features (e.g., Fukishima, 1986; Tsuruta et al 2000).

From the success of the connectionist paradigm, the following properties can be viewed as useful in the design of systems that are required to support low level processing:

1a. The inclusion of discrete layers of processing units that can either be unordered or topographically arranged. Consistent with general connectionist models such as LEABRA, each unit is required to store a real-valued activity level during each discrete time step.

1b. The inclusion of subsymbolic transformations between and within layers that are consistent with the connectionist paradigm (such as passing the weighted sum of the inputs to each cell through a non-linear transfer function).

1c. The ability to define the architecture (i.e. number, connectivity and size of layers) to afford easy modelling of different tasks.

4.1.2 The ability to create and manipulate complex hierarchical representations

Tasks such as planning, problem-solving and reasoning can often be effectively described in terms of the exploration of discrete symbol structures (Newell, 1990). Symbolic production systems such as SOAR and ACT-R have several useful properties that afford the manipulation of such complex relational structures:

2a. The ability to represent problem spaces containing a dynamically changing number of objects, properties and relations.

2b. The ability to dynamically add, delete or modify objects, properties and relations.

2c. The ability to express transformations in terms of symbolic rules (productions). Such rules allow the system to be readily programmed to encode new domains, and are transparent enough to be easily understood.

2d. The ability of productions to utilise variable bindings. Such variable bindings provide a powerful mechanism for object invariant pattern matching.

2e. The ability to retrieve information from long-term memory to allow concept invariant transformations (such as retrieving the successor of a currently stored number to perform counting).

4.1.3 The ability to perform self-watching and mental regulation

During the exploration of a problem space, there are many situations where general heuristics will lead the system down a dead end. In such cases, backtracking and preventing re-exploration of the same path is required. Out of the various UTCs and Fluid Analogy models, Metacat has implemented the most comprehensive account of this aspect of problem solving, with the inclusion of two main properties:

3a. An explicit episodic memory that stores explored problem states (potentially at a high level of abstraction).

3b. The ability to use the episodic memory to regulate resources to avoid re-exploration of the same states.

4.1.4 The ability to process information in a contextually sensitive manner

Context not only plays an essential role in aiding with recognition in impoverished real world environments (Biederman, 1972), but also has been shown to affect high-level cognitive processes such as memory, decision making and reasoning (Kokinov, 1994a). As suggested by connectionists such as McClelland and Rummelhart (1988)

and O'Reilly and Munakata (2000), in order to model contextual dependencies, transformations should be gradual and interactive. In implementing such transformations (as discussed in chapter three), the following representational and transformational properties are typically employed:

4a. The representations should support partial information of potentially inconsistent interpretations. For example, in recognising an ambiguous letter in the context of a word, the several letter alternatives should be partially active to allow the activation of likely word candidates.

4b. The transformations should gradually update the representations over time.

4c. The transformations need to support partial matching of the patterns that they react to (e.g., to activate likely word candidates in the above example).

4d. Incompatible interpretations should compete for resources, so that only one solution will dominate over time.

4.1.5 The ability to select and attend to relevant aspects of the world

During problem solving, due to cognitive limitations, it is often impossible to explore all potential alternatives in parallel. Instead, high-level perception is often considered serial in nature, attending to only one avenue of exploration at a time (Newell, 1980). According to O'Reilly and Munakata (2000, p17), such attention can be modelled intrinsically in connectionist systems in which there is a competition for resources (such as when a k-winner-take-all algorithm is employed). Such a restriction ensures that only the most prominent information is represented and transferred. As contextual sensitivity also requires a competition of resources (property *4d* above), the implementation of attention does not necessarily impose extra constraints on system design beyond those already mentioned above.

4.1.6 The ability to exhibit rational, goal-directed behaviour

According to Newell (1980) and Anderson (1993), humans perform actions in a rational and goal-directed manner. Modelling this ability again does not necessarily

impose design considerations beyond those already mentioned, as with ACT-R and SOAR, rational behaviour can be successfully captured within a production system (covered by properties *2a-e* above). That is, production systems readily accommodate rules that can encode such strategies as means-end analysis that are important in capturing the goal directed triggering of actions. In cases where several incompatible productions are triggered, such systems do however require the use of conflict-resolution strategies. Such conflicts can be resolved through a competition for resources, captured by constraint *4d* mentioned above.

4.2 An Overview of the Architecture of FAE

The architectural and processing properties mentioned above provided tight constraints on the design of the Fluid Analogies Engine. In terms of the general processing modules required however, they are fairly consistent with the Fluid Analogy approach, requiring a working memory (*constraint 2a*), a semantic memory (*constraint 2e*) and an episodic memory (*constraint 3a*). To afford the processing of raw distributed information (*constraint 1a*), FAE augments this general architecture with the addition of optional low-level modules (synonymous with layers in connectionist networks). In transforming the stored information, FAE utilises a connectionist production system that is capable of manipulating complex hierarchical information using symbolic rules (*constraint 2c*), as well as also supporting the subsymbolic transformations required form low level processing (*constraint 1b*). The overall architecture of FAE can be seen in figure 4.1.

4.3 FAE in Detail

4.3.1 Representations in FAE

The selection of appropriate representations to use by FAE was guided by several specific constraints: they needed to support raw distributed information in the form of layers of neuron-like units (constraint *1a*), they needed to support the representation of dynamically changing complex relational structures (constraint *2a*), and they

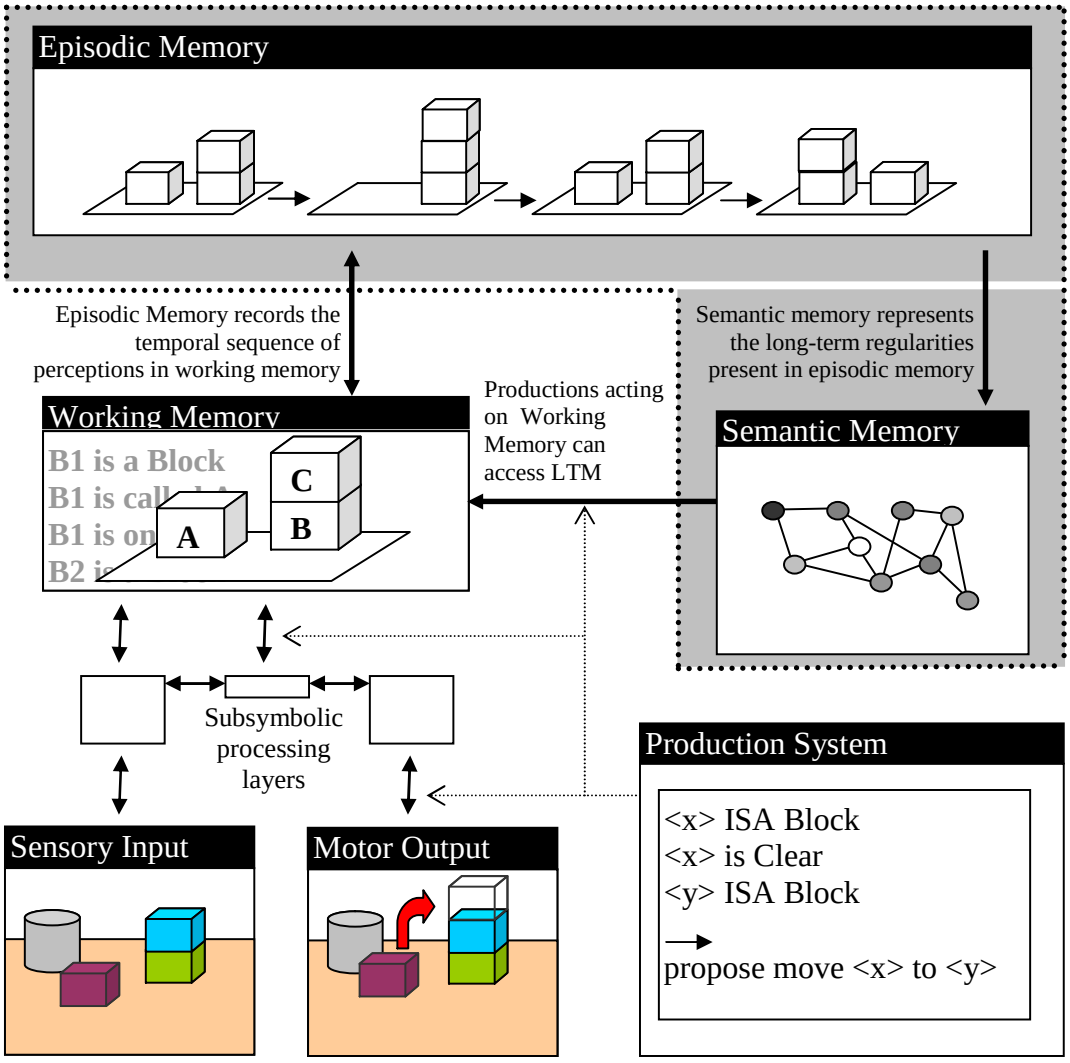


Figure 4.1: The architecture of FAE. FAE contains a Working Memory and a set of subsymbolic sensory and motor processing layers. These representations are transformed through a set of rules defined by the production system that regulate the flow of information, both between and within layers. Information that appears in working memory is temporarily stored in episodic memory, recording the list of states that have been explored during problem solving. The contents of semantic memory reflect the long-term regularities found in episodic memory and are accessible to the production system.

needed to support graded activation in order to allow for a gradual flow of information (constraint *4a*). Although connectionist representations support the first and last constraints, representing and manipulating hierarchical information using a fixed set of units is a complex and relatively opaque task. For this reason, a hybrid approach was devised, merging the types of representations used by the symbolic and connectionist approaches. Specifically, in FAE, as in the symbolic approach, information is represented in terms of objects, properties and relationships (addressing constraint *2a*). As with the connectionist approach however, each such entity is

expressed as a distributed vector (satisfying constraint *1a*), allowing for the representation of either a known symbol (in terms of a set pattern of activity) or the representation of subsymbolic distributed information. These two representational modes are described in more detail in the following sections.

FAE's Working Memory

Working memory in FAE is an architecturally distinct component that is used to represent complex symbol structures involving a variable number of objects, their attributes, and the relationships between them. Similar to other UTCs, each object has a unique name (such as *Table1*, or *Block2*), and a set of attribute type-value pairs (such as *Colour=blue*, *Size=big*). Relationships between objects are also described in terms of type-value pairs, reflecting different aspects of the link (such as *Relative_size=greater_than*, *Relative_location=on_top_of*). Using such structures, FAE is capable of representing the same scope of problems as prominant UTCs such as SOAR and ACT-R.

In order to allow for contextual sensitivity, an interactive flow of information is necessary, requiring that the representations support the partial activation of (and competition between) likely candidates. In FAE, this process is achieved by representing the value assigned to each attribute with a vector of real numbers (e.g, Colour = [0.9,0.1,0]) and by representing known concepts (such as blue and green) as binary patterns (e.g., blue = [1,0,0] and green=[0,1,0]). Using such representations, several concepts can be activated to various degrees simultaneously, where the level of activation can be calculated using the dot product between the current vector and the various concepts. By allowing the production system to dynamically alter the number of vectors corresponding to distinct objects, attributes and relations, complex symbol structures can be represented and manipulated in an efficient and transparent manner (see figure 4.2).

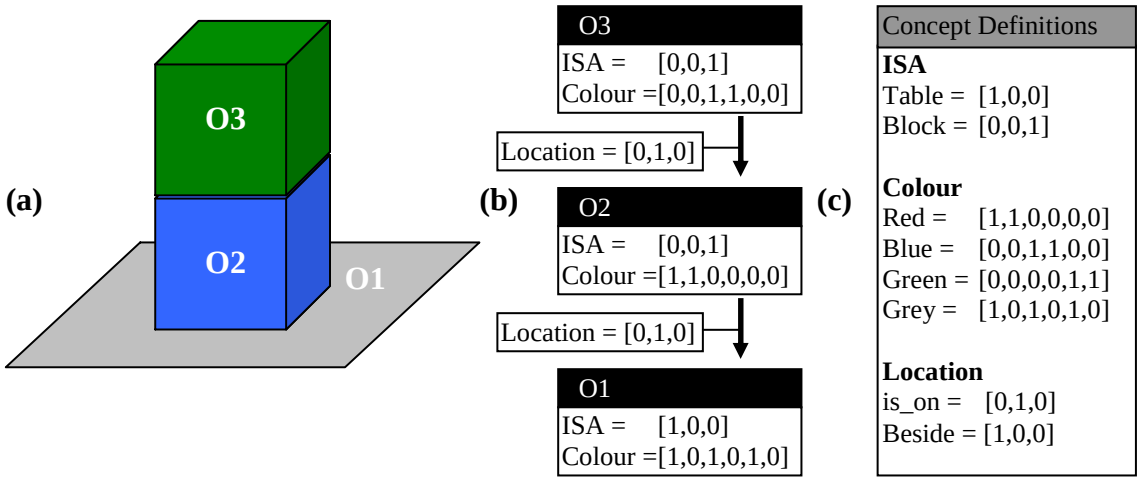


Figure 4.2: The vector descriptions of object properties and relations in FAE. (a) a visual depiction of two stacked blocks on a table. (b) The representation of this situation in FAE, using distributed vectors to represent the attribute and link type values. (c) Known concept vectors provide a symbolic interpretation of the given feature vectors.

Low-Level processing in FAE

As a means of representing object attributes and relations, numerical vectors were chosen as they also afford the processing of raw distributed information. As mentioned earlier, in processing low-level information, a number of discrete layers can be defined in FAE. These layers are useful for encoding the subsymbolic transformations between sensory input and motor output that can occur either directly, or indirectly (through working memory).

The low-level processing layers used by FAE house a one or two dimensional array of objects that can either be treated as unordered or as topographically arranged, being consistent with the general layers of nodes used by connectionism. In the topographic maps of FAE, apart from objects being assigned a set of attributes (such as Colour, in order to represent a visual array), objects are assigned the horizontal and vertical co-ordinates of their position in the grid (see figure 4.3). The relative co-ordinates of objects can then be used by the production system to encode transformations between layers that integrate information across spatially organised clusters (such as encoding local edge detectors for use in visual processing). As with connectionism, the number

and size of the low-level layers used by FAE is user defined, allowing a range of different tasks to be readily modelled.

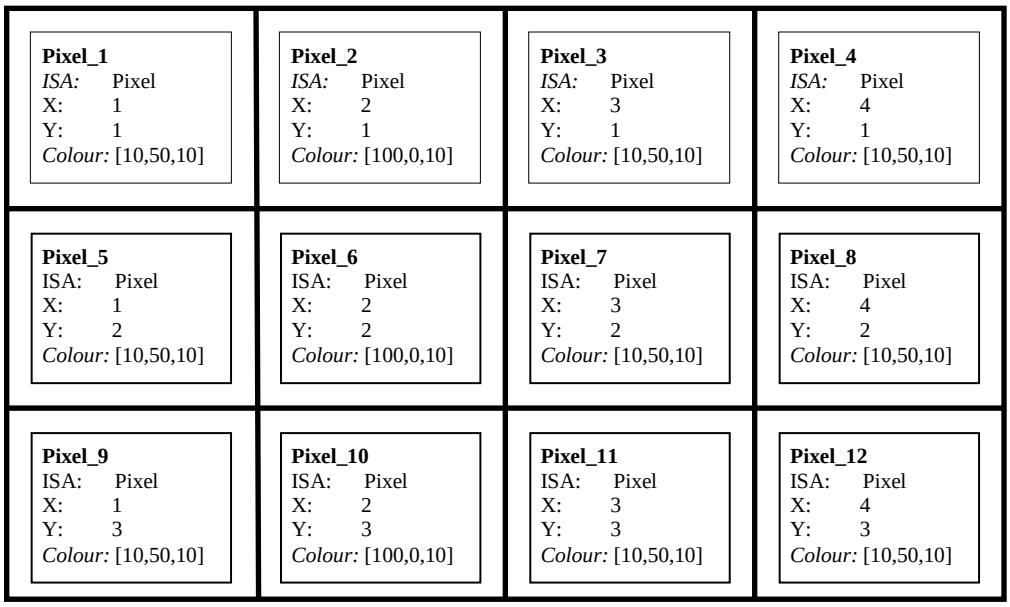


Figure 4.3: An example topographic map in FAE. This example represents a small array of pixels used in processing an image. Each object has an X and Y coordinate as well as an associated pixel colour (expressed in terms of red, green and blue values). The location of objects is known to the production system, allowing transformations based upon groups of spatially arranged features.

4.3.2 FAE's Production System

The development of a means of transforming the representations in working memory and the low-level processing layers was guided by a number of specific constraints: subsymbolic transformations should be consistent with well-known connectionist algorithms such as LEABRA (constraint *1b*); they should support the manipulation of symbolic relational structures using easily defined symbolic rules that allow for variable bindings (constraints *2b-d*); the transformations used should be gradual and allow partial matching (constraints *4b* and *4c*); and there should be a competition for resources within incompatible sets of alternatives to allow the emergence of a single solution over time (constraint *4d*).

Although connectionist approaches provide an excellent mechanism for processing distributed information in a context sensitive manner, they have two main limitations

with respect to the stated constraints. Firstly, most neural networks have a fixed architecture, making them not well-suited to handle the dynamically changing number of vectors in FAE's working memory. And secondly, they do not naturally take the form of transparent symbolic rules that can be readily programmed by the user. The symbolic rules utilised by general production systems are likewise limited in that, although they are able to handle dynamic symbol structures in an understandable manner, they do not readily afford the processing of distributed information or the implementation of contextual sensitivity. For these reasons, a hybrid approach in FAE has been taken, using a symbolic production system that dynamically creates a neural network that can process both symbols and distributed vectors in a transparent and context-sensitive manner.

The productions used by FAE are similar to those used by SOAR and ACT-R, containing a set of preconditions that need to be satisfied for a set of actions to take place. Such productions allow complex symbol structures to be manipulated in an intuitive and transparent manner. In FAE, the specified actions are able to change the values of existing object properties and relations, and in the case of working memory, can add or delete vectors corresponding to new objects, properties or relations. As with SOAR and ACT-R, the productions can be symbolically defined and can contain a set of variables that allow them to be bound to multiple objects or values. For example, if encoding a domain similar to Copycat, a production may be written to identify the "successor" relationship between adjacent letters. In such a case, there may be several instances in working memory for which the preconditions are met that will be bound by different instantiations of the production, allowing successor bonds to form at several locations. The use of variable bindings is a powerful feature of production systems, allowing the same rule to be applied to several positions, preventing the need to specify a separate rule for each individual case.

In order to support subymbolic processing as well as a gradual and interactive flow of information, FAE dynamically constructs a neural network to perform the transformations. This network is formed by creating a "hidden" node for each unique production instantiation linking its corresponding bound precondition and action vectors. For example, in creating a "successor bond" between two adjacent letters, a hidden node would be constructed, connecting the letter vectors to the forming

relationship vector (see figure 4.4). To support the influence of partially active information (required for contextual-dependent processing), nodes can form from production instantiations when the preconditions are only partially matched (above a threshold value).

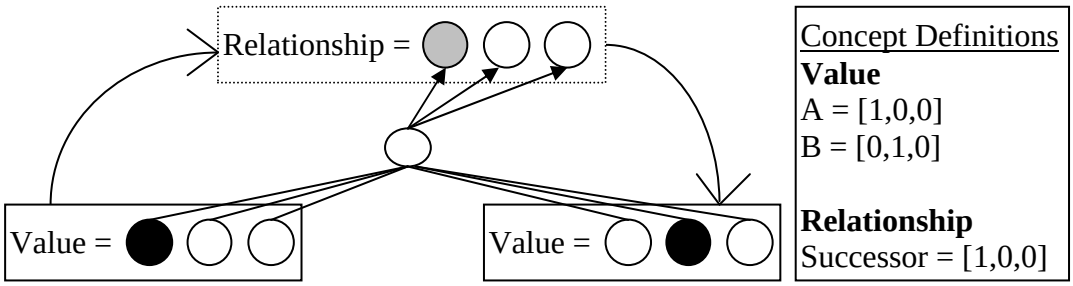


Figure 4.4. The creation of a hidden node to perform a symbolic transformation. In the above scenario, there are two objects representing the Letters A and B. In creating a "Successor bond" between them, a hidden node is constructed to feed the appropriate pattern of activation into the relationship vector.

In some cases in the dynamic construction of the neural network, there may be currently no object, property or relation corresponding to the action arguments. For example, in implementing a program similar to Numbo that solves anagrams, a production may be specified to chunk the letters "T" and "S" into a new object representing the consonant cluster "ST". In such cases, once the preconditions are met, FAE can dynamically create the new object and property vectors during the linking process (see figure 4.5). To prevent an ever-growing increase in the number of such structures, vectors can be destroyed once their activation is at a subthreshold value. Such decreases in activation can occur when the preconditions are no longer met, or as will be discussed, through inhibition from competing production instantiations and through FAE's backtracking mechanisms.

As with connectionism, FAE supports a continuous flow of information from potentially partially active sources. Partial matching of preconditions is used to determine the activation of each of the corresponding hidden nodes, creating various degrees of pressure to perform the corresponding transformations. The partial matching of each precondition of a specific production instantiation is achieved by comparing the distributed vectors associated with the expected and actual attribute values (such as checking if the colour attribute vector of an object is equal to the

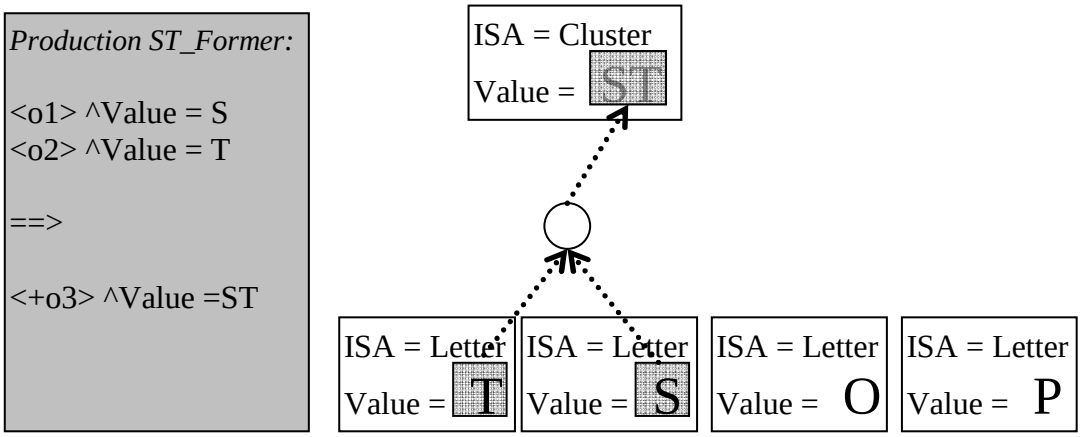


Figure 4.5: The dynamic construction of object property vectors and hidden nodes. (a) Is a production that binds two objects (o1 and o2) with Letter Values "S" and "T" and creates a new object (o3) with value "ST". (b) given a workspace containing the letters T, S, O, P, the given production will bind to the first two objects, and create a new object that has a zeroed Letter Value. A new node will also be created to spread the "ST" pattern to the newly created object.

vector denoting blue). In calculating the degree to which each precondition matches, the dot product of the required and actual values are used (normalised for length), resulting in a match value between 0 and 1. If multiple preconditions are present, the overall match is calculated to be the minimum match of the set of preconditions, implementing a logical AND on the fuzzy values. Preconditions can also be combined using a logical OR, in which case the maximum match of the set of arguments is taken to be the resulting match.

In some cases, a threshold value is set on the precondition matches, so that only production instances with a greater value will be turned into hidden nodes. This procedure is used particularly in working memory, where percepts need to be "conscious" (i.e. above threshold), for them to be mentally manipulated. The overall equation for calculating the precondition match (*P*) of a given production instantiation is given in box 1.

In order to enable contextually dependent processing, all productions in FAE fire in parallel, updating representations in a gradual and interactive manner. This process allows for a natural integration between bottom-up and top-down pressures not found in other UTCs such as SOAR and ACT-R. In using such an approach however, it is

Box 1: The Precondition Match for a given production instantiation.
Given a set of preconditions, the precondition match, *P*, is the maximum of the individual matches, *j*, given by:

$$P = \max_{j \in J} p_j$$

where the individual match, p_j, for condition *j* is given by

$$p_j = \begin{cases} \eta_j & \text{If } \eta_j \geq t_j \\ 0 & \text{otherwise} \end{cases}$$

t_j is the threshold match for precondition *j*
η_j is the calculated normalised match for precondition *j*, given by:

$$\eta_j = \frac{\sum_i x_{ij} y_{ij}}{\sum_i x_{ij}}$$

x_{ij} is the expected pattern vector of precondition *j*, for each vector feature *i*
and y_{ij} is the actual pattern of precondition *j*.

possible that incompatible interpretations will become active (such as activating all possible interpretations of an ambiguous letter). In order to resolve such conflicts, FAE uses LEABRA's update functions to create stable attractor basins within its connectionist network that enforce the emergence of a single interpretation over time. In accordance with models of context-dependent recognition such as McClelland and Rummelhart's IAC model of the word superiority effect (1981), the LEABRA algorithms allow for the temporary initial partial activation of competing alternatives (see figure 4.6) that can be used to activate higher-order representations to help resolve any low-level ambiguities. LEABRA was chosen as the specific approach for implementing interaction and competition in FAE as it is has been successfully used for modelling a large range of low-level cognitive phenomena, in a somewhat biologically plausible manner (as discussed in section 3.1.2). Thus, utilising the same algorithms in FAE, it can be assumed that FAE will also have the potential to model a wide range of low-level cognitive behaviour.

As with LEABRA, attractors in FAE are created through bidirectional connectivity between layers, and the use of global inhibition to promote the activation of only the most salient alternatives. For simplicity, hidden nodes in FAE use a linear activation

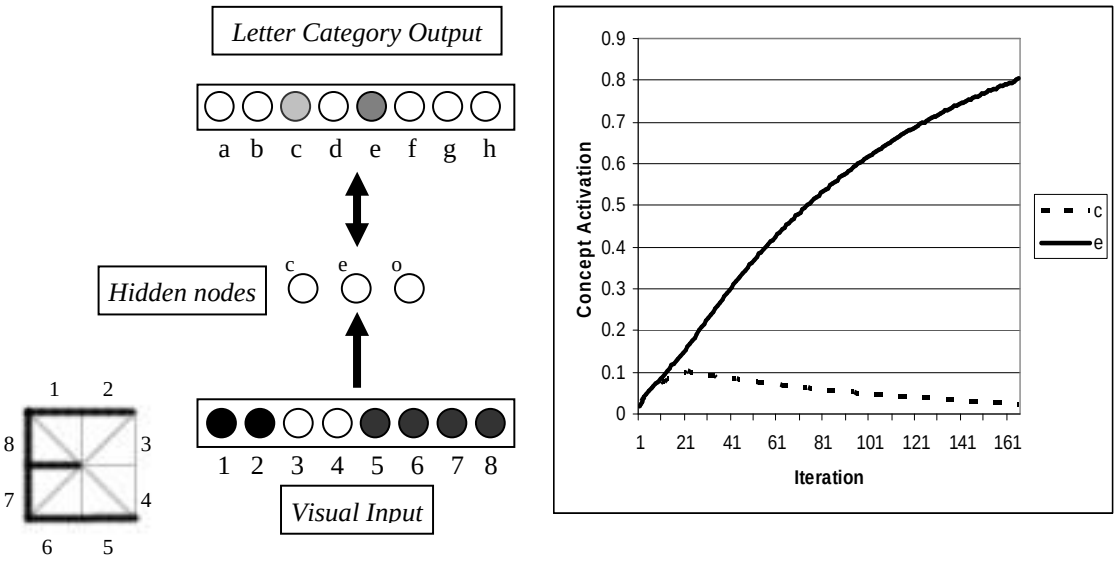
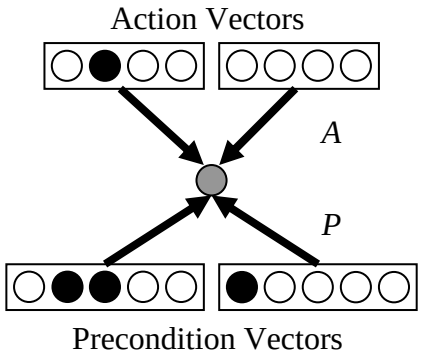


Figure 4.6. Attractor dynamics in FAE. (a) Given several hidden nodes corresponding to likely interpretations of a visual input, activation will spread gradually through the network, partially activating likely alternatives initially, but settling into a single interpretation over time (b).

function (described in box 2), equalling the weighted sum of the match of the preconditions (measured by *P* above) and the match of how close the action vectors are to their target values (denoted by the variable *A* that is calculated in a similar fashion to *P*). At the output layer, FAE uses LEABRA's original update function to update the "membrane potential" by combining excitatory input (the weighted sum of its inputs, integrated over time) with leak and layer inhibition. The level of inhibition for a layer is calculated by using a *k-winners-take-all* (kWTA) algorithm that locates the *k*th and *k*+1th units with the highest membrane potential, and sets the current inhibition level to between these values. The output of a node is calculated by passing

Box 2: Activation of Hidden Units



*T*he activation of a hidden unit (i.e. the instantiation of a production instance), *x*, is a weighted sum of the precondition match *P*, with the action match *A*, given by:

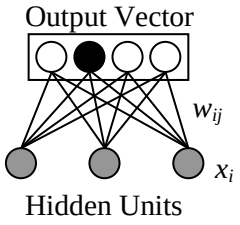
$x = (P+A)/2$

Note: The action match *A* represents how close the current output state is to the target output state for the unit, and is calculated in the same manner as P

As *A* and *P* can range between 0 and 1, the resulting activation of the hidden node will also lie between 0 and 1

the resulting membrane potential through a noisy-X-over-X-plus-1 (noisy XX1) function resulting in a value between 0 and 1. As the XX1 activation function crosses the Y axis at around a value of 0.3, when several competing alternatives share the same level of membrane potential, although their normalised input will be close to 0, they will all gain partial activation. Due to amplifications through the recurrent connections however, small differences between alternatives will be extenuated over time allowing a single solution to dominate. The activation functions used by FAE are summarised in Box 3 below (see O'Reilly & Munakata, 2000 for more detail).

Box 3: LEABRA's Output Activation Functions used by FAE



The excitatory net input (g_e) for a specific output unit equates to the time-averaged sum (using time-constant dt_{net}) of the hidden unit activation values (x_i) multiplied by their corresponding weights (w_i):

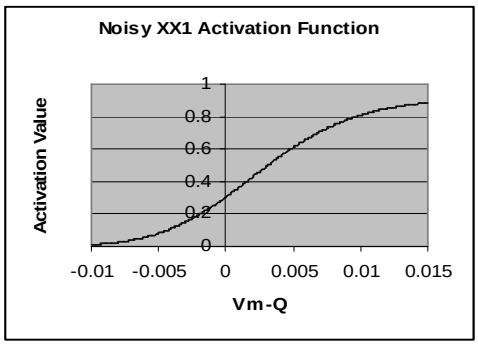
$$g_e(t) = (1 - dt_{net})g_e(t-1) + dt_{net}\sum_i x_i w_i$$

The "membrane potential" (V_m) for a node is calculated from a combination of excitatory input (g_e), inhibition (g_i) and constant leak (g_l), using their resting membrane potentials (E):

$$\begin{aligned} V_m(t+1) = V_m(t) + dt_{vm}[& \\ & g_e(t)\bar{g}_e(E_e - V_m(t)) + \\ & g_i(t)\bar{g}_i(E_i - V_m(t)) + \\ & g_l(t)\bar{g}_l(E_l - V_m(t))] \end{aligned}$$

Layer inhibition (g_i) is calculated using a *k-winners-take-all* algorithm, that sets this value between the membrane potentials for the strongest *k*th and *k*th+1 units.

The output of a given unit is calculated by passing its membrane potential adjusted for firing threshold (Q) through the Noisy XX1 Function (shown opposite).



Competition between Vectors

In many situations in FAE, competing alternatives correspond to the formation of different structures in the workspace, rather than simply differing patterns over a fixed set of output vectors. For example, in solving anagrams, the letters "S" and "T" can be chunked into the consonant cluster "ST" or "TS" that may be represented by different objects in FAE. To allow competition between such structures, productions in FAE can include patterns to identify such conflicts (such as deeming any consonant cluster objects linked to common letters to be incompatible). In such cases, the conflicting structures are linked through inhibitory connections so that only one of the interpretations will dominate over time (shown in figure 4.7). Using such a mechanism, all conflicts (both within vectors and across structures) can be resolved at the local level.

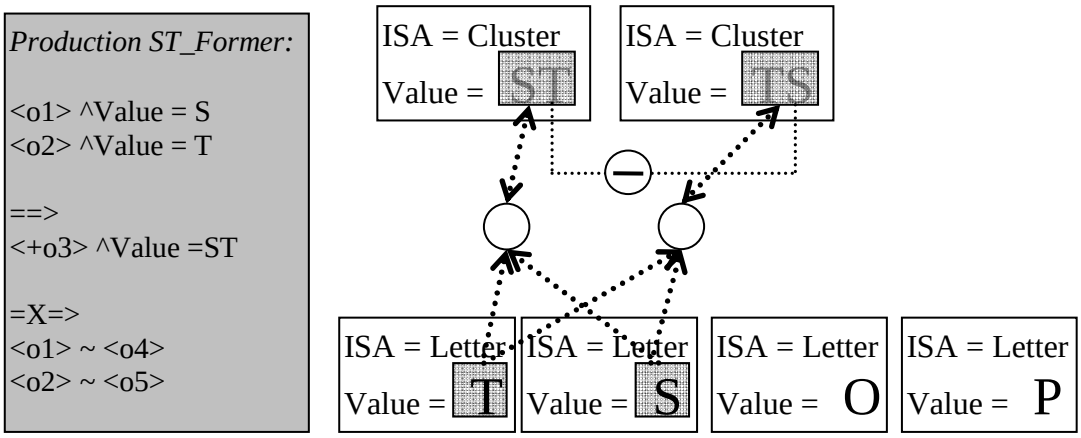


Figure 4.7: The linking of inhibition between incompatible structures. (a) Productions can include patterns that match incompatible structures (such as "<o1> ~ <o4>" that identifies any foreign object linked to <o1>) (b) Incompatible structures (such as the vectors corresponding to ST and TS) will be linked through inhibitory conenctions so that when one structure dominates, the other will be suppressed.

4.3.3 Semantic Memory

The inclusion of a semantic memory in FAE (constraint *2e*) allows for more compact and general rules to be used by the production system. For example, in performing a counting task, a single production can be used to replace a currently stored number by its successor (i.e. retrieving this value from memory), instead of requiring a separate

rule for each different case. Having a separate semantic memory is also beneficial in that a number of different tasks can access the same common store of knowledge. For example, knowledge of the sequence of letters in the alphabet can be used for reciting the sequence on cue, perceiving patterns required for solving letter-string analogy problems (such as *abc→abd*, *ikm→?*), or can be used as a memory aid for retrieving a persons name (by iterating through the possibilities in alphabetical order). Thus, with the inclusion of a semantic memory, new tasks based on already known dependencies can be readily coded.

In FAE's semantic memory, knowledge is stored in terms of *attribute-relation-attribute* triples (such as *three is_the_successor_of two*) that allows the retrieval of one element given the other two (e.g., retrieving the attribute *two*, given the cue *three is_the_successor_of <?>*). High-order predicates can be stored by using an attribute as an index to represent the information as a whole, and breaking the information into distinct slots (denoted by relations). For example, the fact that 2+3=5, can be represented by the triples, *addition_fact_5 OPERAND1 two*, *addition_fact_5 OPERAND2 three* and *addition_fact_5 RESULT five*, where *addition_fact_5* represents the fact as a whole, and *OPERAND1*, *OPERAND2* and *RESULT* representing the slots that are filled by the numbers *two*, *three* and *five*. Using such a representation, missing slot values (such as 2+3=?) can be retrieved by firstly using the given set of cues (*operand1=two, operand2=three*) to retrieve the indexing attribute *(addition_fact_5)*. The returned indexing attribute can then be used to retrieve the missing information (i.e. using the cue *addition_fact_5 RESULT <?>* to retrieve the value *five*).

In order to support the distributed representations utilised by FAE, semantic memory is stored using a rank-three tensor (Halford et al., 1994). In this method, *attribute-relation-attribute* triples are stored by combining their associated concept vector values using the outer product to form a sparse three-dimensional matrix. The matrix of the different facts can then added together to form a distributed weight matrix (M) representing long-term memory. In order to retrieve an attribute or relation given the other two items from an *attribute-relation-attribute* triple, the vectors representing the two cue items are combined using the outer product and combined with the stored

three-dimensional memory matrix using the inner product (shown in Box 4). This function returns a single vector representing the retrieved attribute or relation.

The use of a rank-three tensor for memory storage, apart from naturally affording the use of distributed vectors for representing knowledge, is beneficial in that it leads to automatic generalisation of learned knowledge. For example, if the representations of the concepts *frog* and *toad* are similar, learning that *frogs EAT flies* will infer that *toads* are also likely to *EAT flies*. That is, given the query *toads EAT <?>*, a slightly degraded version of *flies* will be returned (depending on the degree of overlap between the representation of *frogs* and *toads*). Noisy vectors are also likely to be

Box 4: the storage and retrieval of information from Semantic Memory

Storage

Given the following Vectors:

f = frogs, t = toads
e = eat
b = bugs fl = flies

the information that frogs eat flies and toads eat bugs can be stored in the memory matrix (*M*) as follows:

$M = f \times e \times fl + t \times e \times b$

Retrieval (and generalisation)

Given memory matrix *M*, the vector corresponding to the retrieval request *frogss eat <?>* can be calculated as:

$(t \times e).M = (f \times e).(t \times e \times fl) + (f \times e).(t \times e \times b)$

Which will return *fl* plus a degraded version of *b* (if the vectors for *frogs* and *toads* are overlapping).

returned in cases where there are several stored facts that match (or partially match) the given cues (e.g., *frogs* may also *EAT bugs*). Such noisy vectors can be decoded into matching concepts and their corresponding retrieval strengths by comparing them to the known concept vectors (such as the vector corresponding to *flies*) using the dot product.

Productions in FAE treat retrieval requests in a similar manner to preconditions, containing a variable that will be bound to the retrieved alternatives, and using the retrieval strength of each alternative for the corresponding precondition strength. As a result, each different returned alternative will be bound to a different production

instantiation and represented by a different hidden node in the network to create competing attractors of various strengths.

4.3.4 Episodic Memory

FAE's Episodic Memory is used specifically for storing the discrete states that have occurred in working memory during problem solving (*satisfying constraint 3a*) and regulating the system resources to backtrack out of poor solutions (*satisfying constraint 3b*). In FAE, the identification of poor solutions (and the corresponding structures to destroy) is calculated using a similar procedure to Fluid Analogy models such as Numbo. In these systems, structures formed in Working Memory may be inhibited if, over time, they do not act as a precondition for triggering the formation of further structures. Such structures are useful to destroy as they may have inhibited (through competition) other possibly beneficial structures.

Unlike Numbo and Copycat, FAE's inhibition of structures is more long-term, preventing the system from re-exploring poor solutions (as opposed to Copycat, in which the same structure can reform almost immediately). Inhibition in FAE is also context-dependent, allowing the same structure to be explored when in combination with different alternatives. For example, in solving the anagram *h-c-l-i-d*, FAE may initially chunk the letters into the consonant cluster *ch* and the syllable *lid*. Such a state space would represent a dead end, as they could not be further combined, requiring that one or both structures be destroyed. If *ch* was permanently inhibited however, the correct solution *child*, could not be formed. Instead, *ch* is only inhibited when "*lid*" is present in Working Memory, allowing its re-emergence when the letters *l-i-d* are rearranged into the rhyme "*ild*."

To support context-dependent inhibition, a form of episodic memory is used that stores the contents of Working Memory in terms of a set of discrete episodes. Each episode corresponds to a unique combination of concepts that are active above a specified threshold. On each iteration the episode corresponding to the current state of working memory gains activation by a fixed amount. Once a threshold value is reached, it is determined that no new structures are likely to form from the current state (i.e. an impasse has occurred), and that structures within Working Memory

should be destroyed. Rather than all structures being inhibited equally however, a single production instantiation is targeted (the one with the most competitors), resulting in the decay of its corresponding structures. Such selective inhibition allows the system to iterate through the different range of solution states, backtracking when impasses are reached.

According to the above description, FAE will avoid previously experienced impasse situations due to its "perfect memory." However, it is possible in FAE to model a less optimal search pattern by specifying a decay term, so that episode strengths decay over time, getting destroyed if they reach below a threshold value. Apart from preventing an ever-growing number of remembered states, episodic memory decay can result in the re-exploration of dead-end salient states, which may better mimic the search patterns exhibited by humans (i.e. it is likely that humans will re-examine previously explored states when they get stuck). However, how much decay results in the best approximation to human performance remains an unexplored question, lying outside the scope of this current thesis.

4.4 Summary

The Fluid Analogies Engine represents a new hybrid architecture explicitly designed to provide mechanisms for modelling both high and low levels of processing. Within this chapter, a set of design constraints were specified, detailing successful mechanisms that have been previously employed by disparate models to capture core cognitive abilities. As described, FAE combines a number of approaches to conform to these constraints. For example, all information is represented as distributed vectors that are manipulated through a connectionist production system that dynamically constructs a neural network from a set of symbolic rules. This approach affords the processing of both raw distributed information and complex structural information in a flexible and context sensitive manner. The ability of FAE to capture each of the six cognitive abilities mentioned in chapter one is assessed and discussed in the following chapters.

Chapter 5

Modelling Contextually Sensitive Processing

Context is often an essential cue for resolving ambiguity in perceptually challenging conditions and for choosing appropriate actions to perform. In some cases, such contextual factors can be expressed as symbolic rules that are appropriate for unified theories of cognition such as ACT-R and SOAR. For example, while playing chess, moves that are generally valid can be classified as invalid if it places the player in checkmate. However, one of the strong tenets of this thesis is that in most real-world environments, contextual sensitivities cannot be modelled using such rigid all-or-nothing symbolic rules. Rather, robust perception emerges out of the subtle interplay between bottom-up and top-down factors that can only be modelled at a subsymbolic level.

Within this chapter FAE's ability to model contextually sensitive processing is explored. Firstly, to highlight the importance of this ability in all levels of cognition, the main empirical studies that have demonstrated such effects are overviewed. The properties required to model this ability are then revisited (from chapter three), discussing how these features are included in FAE. Using the word superiority effect as a testbed, FAE's ability to process information in a contextually sensitive manner is then demonstrated. This chapter concludes with a discussion of the generality of the approach.

5.1 Context Dependencies in Human Cognition

Perception is the act of interpreting the raw data extracted by the sensory organs from the outside world. Such interpretations are required for an organism to choose actions that are appropriate to the situation at hand and are beneficial in evolutionary terms. For an organism to function in real time, the process of perception must be fast and reliable. However, many sources of variability and noise affect the quality of the sensory information, making perception inherently difficult. For example, in the visual recognition of objects, sources of variability such as changes in pose, size, illumination, and occlusion make it unlikely that the sensory information will exactly match anything previously experienced (Ullman, 1998). Thus, any form of extra information that can help resolve ambiguities or facilitate recognition is of great benefit. One such source of information is context; objects and events are grounded in contexts that can often provide evidence that can help interpret the ambiguous and impoverished data given by the sensory organs. For example, a white blob may be interpreted as a telephone given its spatial location in an office, or an ambiguous handwritten character may be interpreted correctly given likely word candidates that match the remaining letters or sentence context.

The effect of context upon perception has been widely studied in the psychological literature in the areas of language, music, and visual perception, dating back as far as psychologists could present the requisite stimuli under controlled conditions (e.g. Bagley, 1900; Cattell; 1886). More recently, technological advances have allowed for more sophisticated means of studying contextual influences. For example, De Graef, Dominiek & D'Ydewalle (1992) studied contextual effects on visual recognition by measuring the duration that people fixated on objects during free exploration of real-world visual scenes. They found that with increased exploration of a scene, durations decreased significantly (indicating faster processing) for objects that were consistent with the context, but increased (indicating slower processing) for objects that were improbable within the current visual scene or were located at unexpected spatial locations. Thus, this experiment provides a strong demonstration that context directly affects the ease of perceptual processing in visual object recognition.

Contextual effects on perception have also been widely explored in language processing (e.g., Warren 1970; Drenth & Ruber, 1997; Pillsbury, 1997). One of the seminal experiments in the area was conducted by Reicher (1969) who demonstrated that subjects are more accurate at recognizing letters that appear in words compared to letters in isolation or in unpronounceable nonwords (the *word superiority effect*). Such a contextual effect on language perception has been found at a number of levels, with the perception of words themselves being enhanced when placed in legal sentence contexts (Jordan & Sharon, 2002).

Apart from in perception, the effects of context have also been demonstrated in higher level cognitive processing. For example, in memory studies, retrieval of particular facts has been shown to be influenced by the phrasing of the question (Kokinov, 1989), and whether or not the external environment was the same as when the fact was learned (Godden and Baddeley, 1975, 1980). The wording of a question has also been shown to have an influence on decision making, affecting the perception of the problem, the evaluation of probabilities and the resulting decision that is made (Tversky and Kahneman, 1981). Context has also been demonstrated to play a role in problem solving, where accidental objects or events can influence reasoning processes (such as the role of the falling apple in inspiring Newton's theory of gravity) (Kokinov, 1996).

The studies mentioned above demonstrate that contextual sensitivity is ubiquitous in thought, ranging from low-level subsymbolic tasks, such as object recognition, to high-level cognitive tasks such as language comprehension, memory, decision making and reasoning.

5.2 FAE and the Modelling Requirements for Context Sensitive Perception

According to McClelland and Rumelhart (1981), there are four basic properties of cognition that support the processing of information in a contextually sensitive manner (as explained in chapter three):

(1) Perception occurs in a multilevel processing system

(2) Deeper levels of processing are accessed via intermediate levels

(3) *Processing is interactive* (both bottom-up and top-down)

(4) Information flow is continuous

In accordance with this view, O'Reilly and Munakata (2000, p14-18), cite *parallelism*, *gradedness* and *competition* as being central properties of cognition that support interactivity and a continuous flow of information. That is, *gradedness* allows for partial activation of alternatives, with *parallelism* being used for a continuous flow of information. *Competition* for resources is also required in order to resolve conflicts and ambiguities in interpretation.

In order to model context sensitivity in FAE, the algorithms underlying information processing were based around the connectionist framework of LEABRA that supports all of the above properties. Firstly, FAE allows a number of different processing modules to be defined (such as a collection of topographic maps) to afford a natural separation of processing layers. For example, in modelling *the word superiority effect*, a separate topographic map can be defined for letter features, letters and words. Alternatively, this segregation can be accomplished by storing all layers within the same module, but utilising different chunk types to segregate representations (such as having letter feature chunks, letter chunks and word chunks). In either case, representations can be easily segregated with FAE.

Secondly, information flow can be defined so that deeper levels of processing can only be accessed via intermediate levels rather than from the raw input. As activation flow is achieved by productions, regulating the flow of information is achieved by specifying productions that make the required conversions (such as specifying productions that activate letters from letter features, and words from individual letters). In the same manner, the third property underlying contextual sensitivity, namely interactive processing, can be achieved by defining productions that process information in a top-down manner, creating a cyclical flow of information.

As stated in the previous chapter, information flow in FAE is continuous, satisfying the fourth property specified by Rumelhart and McClelland. In this model,

productions do not fire in an all-or-nothing fashion, but rather, to various degrees, based upon the degree of match of the preconditions. Generally, a threshold is set so that productions will only fire when enough evidence is present. However, in modelling phenomena such as the word superiority effect, this threshold can be set to 0, resulting in all productions firing at each iteration (supporting *parallelism*), but with each contributing to the outcome with a strength dependent on the match of their preconditions to the current state. As with LEABRA, *gradedness* is supported by FAE by allowing nodes to take varying levels of activation, with a *competition* for resources occurring though global inhibition that regulates the amount activation present at each layer. This general strategy allows for a continuous flow of activation that can be used to transmit information in a bidirectional manner, resulting in a strong interaction between bottom-up and top-down factors required to model context dependent processing.

5.3 The Word Superiority Effect

Contextual effects in language processing were discovered over 100 years ago, with one of the early findings being that subjects could identify more letters in words than in random strings that were flashed briefly on a screen (Bagley, 1900). However, from such experiments it was not clear if this was a true contextual effect, or if it was more due subjects' chunking the information (with subjects being able to decode the letter constituents of words when needed). Such strategies were controlled for in an experiment by Reicher (1969) that was still able to demonstrate a perceptual superiority for letters appearing in words. In this experiment, a word, an unpronounceable non-word, or an individual letter was briefly shown to the subject, followed by a mask of random letter features. With the presentation of the mask the subjects were given two letter alternatives to choose between that were presented above and below where the critical letter appeared. Guessing was controlled for by giving alternatives that both generated valid words in the word condition. For example, given that the presented word was *boat*, and the third letter was to be identified, the letters *o* and *a* would be presented as alternatives (i.e. above and below where the third letter appeared) to form the valid words *boot* and *boat*. In this experiment, subjects were more accurate at choosing the letter that appeared in a word than in an unpronounceable non-word or isolated letter (i.e., *the word superiority*

effect). Such an effect seems paradoxical, as to recognise a letter in isolation, only the letter needs to be processed, whereas to identify a word, all the letters need to be processed. Thus, intuitively, it would seem that a letter in isolation should be easier to process, which is contrary to the empirical observations which demonstrate that context has a facilitatory effect on perception. Using similar paradigms, subsequent experiments have also demonstrated that the supposed "word superiority effect" can also be achieved using pronounceable nonwords (Johnston & McClelland, 1973; Wheeler, 1970), thus generalising the effect to word-like contexts. The ability of humans to generalise beyond what has been experienced is an essential property of human perception, thus making *the pseudoword superiority effect* (the preferential processing of letters in regular nonwords compared to irregular nonwords or individual letters) also an important phenomenon to be modelled.

5.3.1 The IA Model of the Word Superiority Effect

As stated, one of the most comprehensive models of the word superiority effect is the interactive activation (IA) model implemented by McClelland and Rumelhart (1981). In this model (described briefly in chapter three), there are three distinct layers of processing: the feature level, the letter level and the word level. At the feature level, letters are represented in a distributed fashion in terms of the presence or absence of 16 line features (figure 5.1), with one pool of units being used to represent each of the four letter positions. Within each set of features, there is one unit for representing the presence of a feature, and a separate unit for representing the absence of a feature.

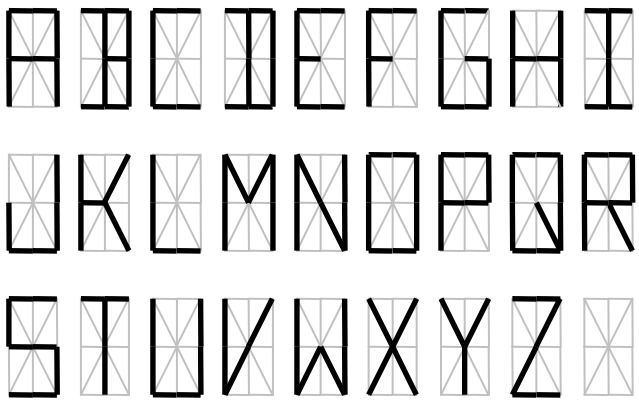


Figure 5.1. The font used to represent letters in the McClelland and Rumelhart IA model of the word superiority effect (1981).

Such a representational scheme allows for encoding that a feature is present, absent, and unknown (such as in the case of impoverished stimuli). In contrast to the feature level, at the letter and word level, concepts are represented using a local coding scheme, where each node represents a distinct letter or word. At the word level, there are 1179 nodes, each representing a different four-letter word from the Kucera and Francis (1967) word list. At the letter level, there exist four sets of letter units, one set for each letter position, each containing nodes representing the 26 letters of the English Alphabet.

In the IA model, activation gradually flows between the nodes between various layers. Firstly, from the letter features that are known to be present or absent at each of the letter positions, activation flows to the letter level. As there is a degree of overlap between letters, several letter nodes may become initially active. However, due to the inhibitory connections between letter nodes, as the strongest match gains activation, it inhibits the other alternatives to a greater degree, gradually settling into a single decision. During this process, word nodes also gain activation from the partially active letter nodes. Similar to the letter layer, the word layer also contains inhibitory connections, so that over time, a single word may prevail. The word layer also contains feedback connections to the letter layer, being responsible for the word superiority effect, with feedback from the most active words (i.e. the words that match the most active letters), facilitating the processing of the letters that they contain. In the case where the input contains a random letter string or a single letter, the letter nodes will not be substantially active, thus not facilitating the processing of individual letters. In the case of pronounceable nonwords however, there may be several word nodes that gain activation due to a high degree of similarity to the nonword. For example, if the non-word TEAD, was presented, the words TEAM, TEAL, READ and BEAD would become partially active, reinforcing processing of the vowels. Thus, pronounceable non-words, rather than activating an individual word unit, will activate a group of word units that as a whole will place top-down pressure on the letter level to reinforce any regularities.

The IA network was implemented to emulate experiments such as the ones conducted by Reicher (1969). In the most simple of the simulations, given two alternatives for a letter, after a set number of iterations of the IAC algorithm, the response probability

was calculated based upon the activation of the corresponding letter nodes (McClelland & Rummelhart, 1981). Such a calculation was shown to display a benefit for letters contained within words and pronounceable nonwords compared to random strings or letters in isolation, thus capturing the main effects of the word superiority effect.

5.4 Modelling the Word Superiority Effect in FAE

The following sections demonstrate the ability of FAE to model contextual sensitivity using the word superiority effect as a test-bed. The representations utilised in this implementation are the same as Rumelhart and McClelland's (1986), using three distinct layers of information. In the FAE implementation, the separation between letter features, letters and words is achieved by using three distinct topographic maps (see figure 5.2). The first two maps contain 4 objects each, relating to the letter features and letter categories perceived at four sequential locations (allowing for words containing four letters to be identified). In contrast, the final topographic map contains a single object representing the perceived word. At the letter feature level, objects are assigned a *letter feature* value, described by a distributed vector of 32 features (one feature for representing the presence and absence of each of the 16 line segments of which the font was constructed). At the letter level, each object stores a *letter category* property that represents the active letters at each time step. This property is represented by a vector of 26 values, using a local coding scheme for each letter, allowing several alternatives to be partially active at a time. At the final word level, words are also represented using a local coding scheme, with all four letter words from the Kucera and Francis word list being included (2317 words in all). Overall, the resulting feature vectors and segregation between layers is identical to the Rumelhart and McClelland model. It is important to note that this example domain represents fairly low-level processing, not requiring any of FAE's high-level architectural features, such as its working memory and episodic memory. That is, it is viewed that word recognition is a relatively automatic process, not requiring "mental deliberation."

In FAE, the flow of information between chunks is regulated by productions. In the case of the word superiority simulation, only three productions are required: a production that feeds information forward from the feature level to the letter level, a production that feeds information forward from the letter level to the word level, and a production that feeds information backwards from the word level to the letter level.

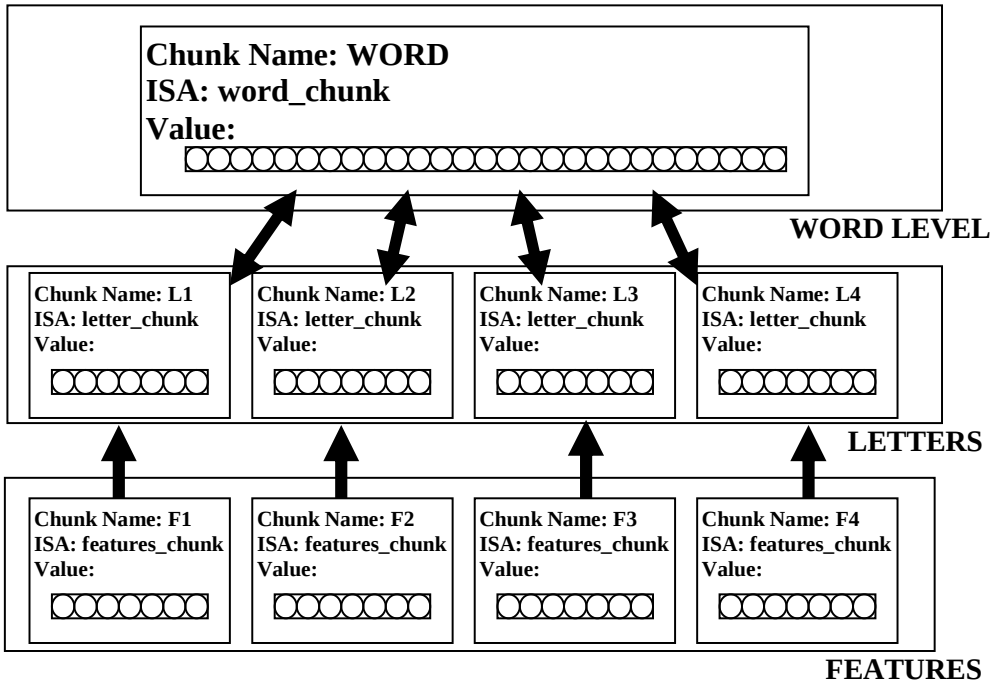


Figure 5.2. The representations used by FAE in modelling the word superiority effect. In this implementation there are three distinct layers, with the first two layers containing four objects storing letter specific representations for each of the four letter positions. At the final layer, a single object stores the distributed vector for the active word. Each object holds a "value" slot that stores a distributed representation of the features or concepts that are active at each level. The arrows represented the flow of information between chunks (regulated by productions).

Although a single production is generally required for each individual mapping (such as a different production for activating the letters *a* and *b* when the corresponding letter features are detected), due to the large number of productions that would be required to define the flow of information for all 2317 words, a shortcut was designed in the scripting language so that only a single production needed to be written to handle similar types of mapping. In this shortcut, a filename can be specified that contains the symbolic values that should fill each of the slots of the production, with one unique mapping being presented per line. For example, in defining a production that maps the letters to an individual word, each row in the file may contain the individual letters and the word that they correspond to (illustrated in figure 5.3).

As described in the previous chapter, productions in FAE do not contain all-or-nothing conditions like SOAR and ACT-R, but rather, there can be variable degrees of matching, with the match being calculated as the dot product between the required feature vector and the actual bound value. If this match is above a defined threshold,

```
Production recogniseWords (patternFile ="WordSuperiority\wordRules.txt"):
   Module: Letters
       <0,0> ^Letters==$1 (threshold=0.0)
       <1,0> ^Letters==$2 (threshold=0.0)
       <2,0> ^Letters==$3 (threshold=0.0)
       <3,0> ^Letters==$4 (threshold=0.0)
  ==>
   Module: Word
       <0,0> ^Word = $5
(a)

      a b b e abbe
      a b l e able
      a b l y ably
      a c h e ache
      ...
      z e s t zest
      z i n c zinc
      z i o n zion
      z o n e zone

(b)
```

Figure 5.3. A shortcut in the scripting language, allowing all forward transformations between the letter and word level to be captured in a single rule (a) the production defining the mapping from letter instances to words, using map position to identify the corresponding objects. The values that are matched are the rows 1-4 ($1-$4) of the specified data file, with the action trying to form the word in the fifth column of the data file. (b) the data file containing the individual letters and words that are to be mapped.

the action will be performed, creating attractors within specified slot values to modify their contents. The strength of the pull of the created attractors are determined by the how well the preconditions are met, with each attractor possibly having to compete with stronger alternatives. For example, in the word superiority simulation, given that there is a high degree of overlap between features of individual words, many attractors will become active at the letter level to varying degrees (depending on how closely the features match the expected values) with initially, all such matching letters gaining activation. As the strength of an attractor is also determined by how close the current pattern of activity is to the attractor basin, as the best interpretation moves

closer to its attractor, the force of attraction increases. As a result, over time, a single interpretation dominates and suppressed the alternatives (see figure 5.4).

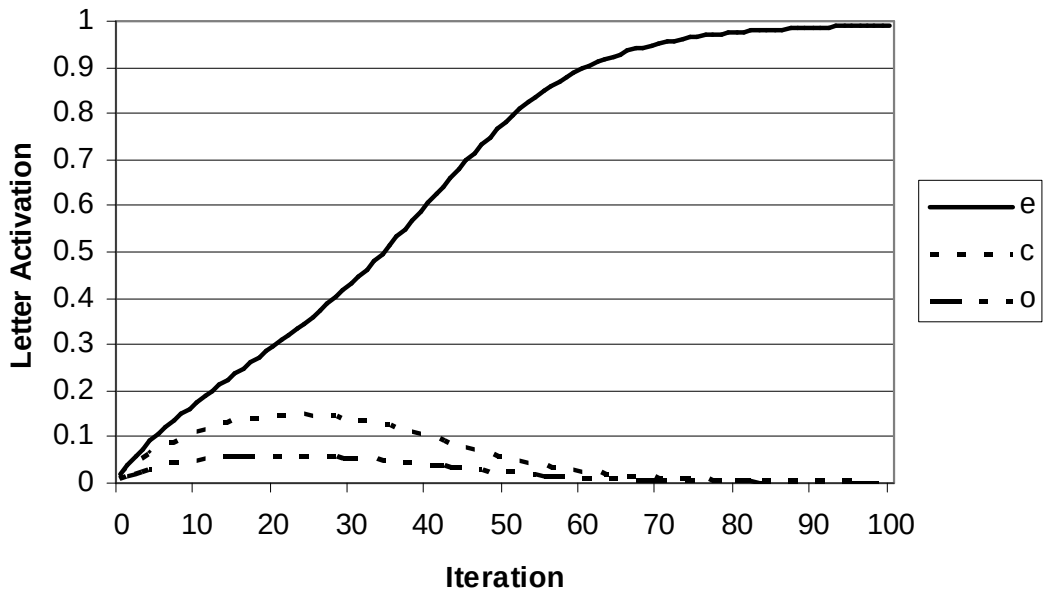


Figure 5.4. The activation of a letter in isolation. Given the letter features of "e", all partially matching letters gain activation in the short term. However, as the activation of the correct interpretation gains activation, and moves closer to its attractor, the force of attraction increases, suppressing the other alternatives.

In the word superiority simulation, as the letter nodes gain activation, information spreads to partially matching words. As many letters are initially active, there are also a large number of partially active words. Over time however, as the letters become better perceived, the more likely word candidates gain the most activation. If the letters form a valid word, this interpretation will dominate over time, suppressing all alternative competitors (see figure 5.5).

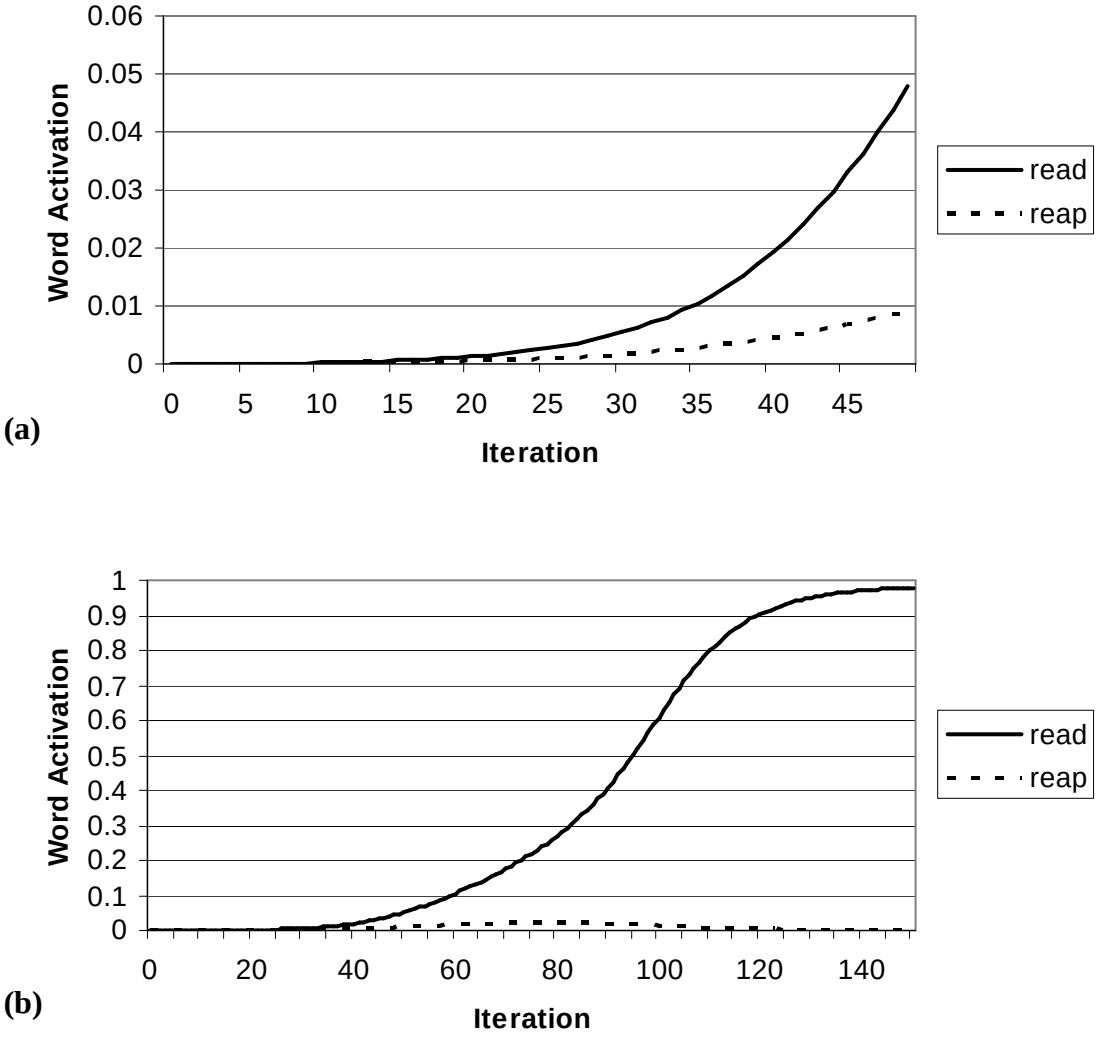


Figure 5.5. The activation of words over time. (a) Initially many partially matching word candidates become active. (b) the correct word interpretation becomes dominant over time, suppressing the other alternatives.

In the word superiority effect simulation, as word nodes become active, top-down pressure is applied to facilitate the processing of the individual letters. However, as stated, initially there will be many such active word nodes, all having an influencing effect. This is particularly true in the cases where the letters form a word or a pronounceable nonword. In both cases, many word nodes that are similar to the input will become active. For example, if the input word is “dead”, other word nodes such as “read” and “deed” will also be initially active, reinforcing the processing of the letter “e” in the second position. For this reason, both words and pronounceable nonwords can facilitate processing at the letter level. The FAE implementation

demonstrates a superiority for words and pronounceable nonwords over random strings, and individual letters (see figure 5.6 below). Thus this implementation, with its gradual and interactive flow of information, replicates the findings of the McClelland and Rumelhart model (1981) displaying local context effects.

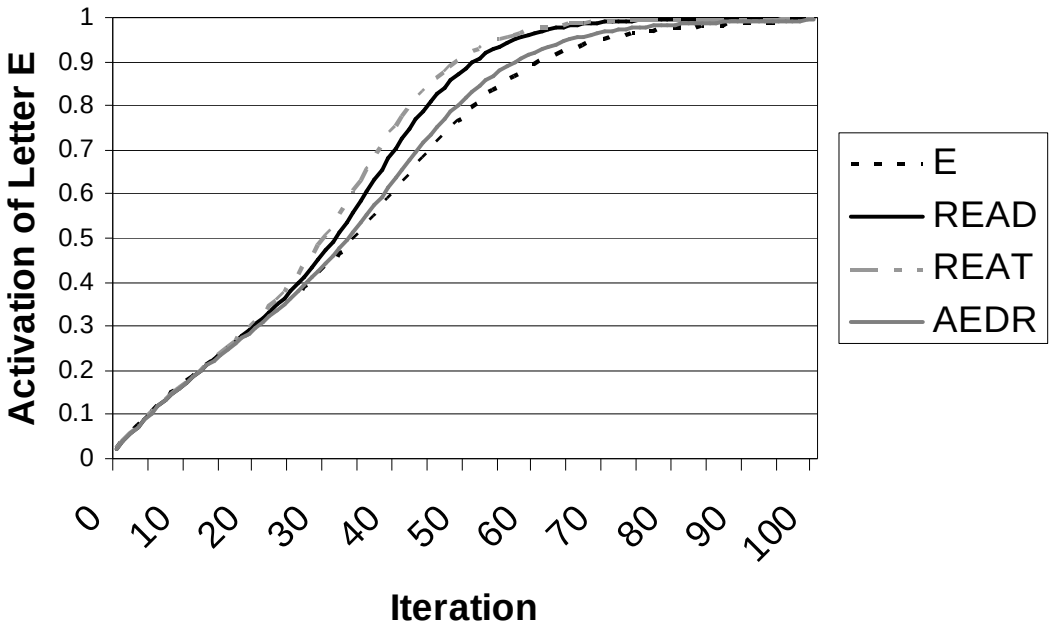


Figure 5.6. The word superiority effect in FAE. Letters (e.g., E) in words (READ) and pronounceable non-words (REAT) are processed faster than in isolation (E) or in random strings (AEDR). This effect is due to the top-down flow of activation from the word level, facilitating the processing of partially active letters.

5.5 Modelling the Resolution of Ambiguity in FAE

In noisy real world environments, context plays an important role in perception, aiding with the interpretation of impoverished stimuli (as discussed in the introduction to this chapter). Using the same representations as the word superiority effect simulation, the use of context to help resolve ambiguities can be readily demonstrated in FAE. In this network, at the sensory level, letters are represented as distributed vectors of features. Ambiguity can arise if a noisy feature vector is presented to the system that is equally close to two known letters. In such cases, as in the word superiority effect, partial activation of word candidates can place top-down pressure at the letter level, helping to resolve the ambiguity. For example, an ambiguous feature vector that can either be interpreted as an A or a P, will be interpreted as a P when

placed in the word HEL**P**, and will be interpreted as an A if placed in the word RE**A**D (as shown in figure 5.7).

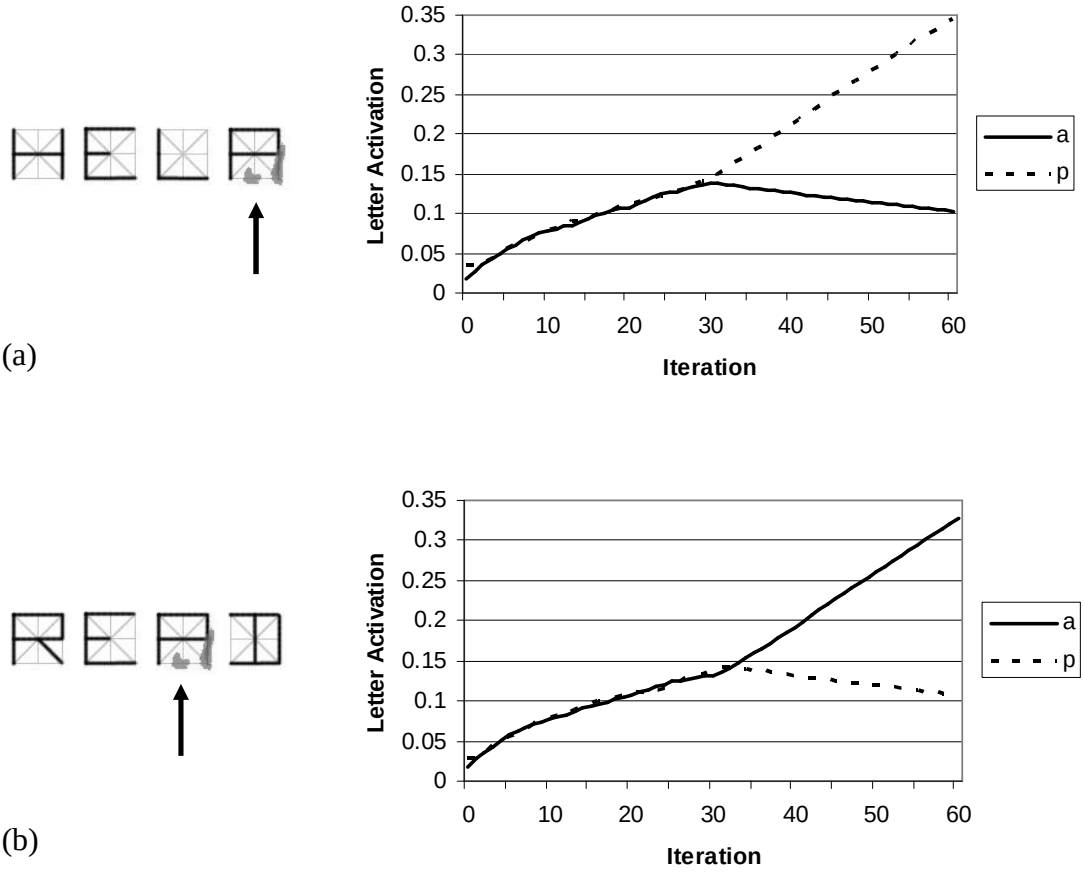


Figure 5.7 The context dependent resolution of ambiguity. An ambiguous letter (A/P), can be resolved in the context of a word, through top-down activation of word candidates. (a) The ambiguous letter is perceived as a "P" in the context of the word "help" (b) The ambiguous letter is perceived as an "A" in the context of the word "read"

5.6 Discussion

One of the central aims of FAE is to capture the general mechanisms underlying contextual sensitivity. The solution employed by FAE is consistent with connectionist models such as LEABRA, utilising distributed vectors that allow several alternatives to be active to various degrees, and using productions that allow for a gradual flow of information in both a bottom-up and top-down manner. These features allow for an interactive and continuous flow of information that can readily be used to model the word superiority effect; an archetypal instance of contextual

sensitivity. Such an example requires a subtle interaction of partially active concepts in both a bottom-up and top-down manner; a property not easily emulated by symbolic systems.

Context effects in FAE (as well as in neural networks in general), can be viewed as differences in the internal state of the system affecting the processing of local information. In the word superiority effect, such influences are top-down, with partially active words influencing the processing of individual letters. However, contextual influences can also occur through lateral connections (i.e. within layer connections), or differences in the internal state upon processing (e.g., temporal contingencies). Neural networks consistent with the graded interactive processing algorithms used by FAE have been successfully shown to model context effects resulting from all such factors. Such examples include semantic priming (Cree, McRae & McNorgan, 1999; Plaut 1995), the calculation of disparity in stereoscopic vision (Marr and Poggio, 1979), resolving figure-ground perceptions (Veccera & O'Reilly, 1998), and generating consistent sets of mappings in analogy-making (Kokinov, 1994a; Holyoak & Thagard, 1989). Such networks are consistent with the approach taken by FAE described in this chapter, evolving a solution through the gradual interaction and competition between alternatives. Thus, the approach taken by FAE in modelling context sensitivity is widely utilised, accounting for a broad range of cognitive phenomena. The ability of FAE to perform complex parallel constraint satisfaction is further demonstrated in chapter eight in an analogy-making task.

5.7 Conclusion

This chapter explored the mechanisms underlying contextually sensitive perception; one of the six core cognitive abilities required for intelligent behaviour described in chapter one. According to McClelland and Rummelhart (1981), four main properties are required by a system in order to display this ability: multi levels of processing, deeper levels being accessed via intermediate levels, and an interactive and continuous flow of information. FAE utilises the connectionist update algorithms of LEABRA that conform to these constraints, utilising distributed representations and

gradual flow of information. The ability of FAE to perform contextually sensitive perception was demonstrated in this chapter using the word superiority effect and perceptual ambiguity as examples. The generality of this approach was also discussed, concluding that the mechanisms employed by FAE offer a parsimonious explanation of context dependent processing at a wide range of cognitive levels.

Chapter 6
Modelling Planning

In humans, important goals are often impossible to achieve in terms of a single action, but instead require a sequence of actions or a set of nested subgoals (Newell & Simon, 1972). In such cases, planning (i.e. the mental exploration of a problem space) is required to determine the set of actions that will lead to the fulfilment of higher-order goals. This chapter describes an application of FAE to the game of Numble (described in chapter two), a domain representative of a larger set of problem solving (or "planning") tasks in which a set goal needs to be achieved from the current state through a series of nested operations.

In the game of Numble, the aim is to construct a given number through the addition, subtraction or multiplication of a set of five smaller numbers. Although the domain of Numble is small enough so that the entire problem space could be searched in finite amount of time using a common personal computer, such a brute force approach is not scalable to larger, real-world domains in which there are an unrestricted number of possibilities. For example, for a person whose current goal is to travel to work, there are an unlimited number of actions that they can perform (such catching a bus, or non-task related actions such as staying at home to read the paper), some of which are more beneficial than others in achieving the original goal. The implementation described in this chapter is inspired by the Numbo project (Hofstadter & FARG, 1995) that was aimed at modelling the heuristic nature of human problem solving that allows a rational search to be performed by biasing the selection of operators to the ones most likely to be beneficial in the current situation. However, in the game of Numble, as with any real-world problems, no heuristic search is perfect, with the system likely to be lead down a dead-end on occasion. Thus, apart from requiring a

mechanism to select reasonable operators, a form of backtracking is also needed escape impasses.

This chapter details an implementation of Numbo, demonstrating how planning can be achieved within FAE utilising a number of the cognitive abilities mentioned in chapter one. This chapter begins by listing the abilities that are required for solving Numble problems, followed by a description of the FAE implementation. The generality of the approach to problem solving is then discussed.

6.1 Cognitive Processes Required

In formulating a solution to Numble in a cognitively plausible system, four of the six abilities mentioned in chapter one are required. These abilities are as follows:

6.1.1 Ability 4: Creation of complex hierarchical representations

In the Numble domain, solutions are formed by combining bricks through addition, subtraction and multiplication to "build" the target number. Each operation results in a new number (called a block) than can be further combined. Thus, the solution to a Numble problem can be viewed as a potentially complex hierarchical representation (see figure 6.1).

6.1.2 Ability 2: the ability to select and attend to relevant aspects of the world

In Numble, as in many planning tasks, there are too many possibilities to explore in parallel. For example, given five bricks, there are over 300 different ways that the numbers can be combined through addition, substraction and multiplication. In the implementation detailed in this chapter, as with the original Numbo (Hofstadter & FARG, 1995), it is assumed that humans are likely to attend to only one such problem state at a time, exploring the alternatives in series. Thus a form of attention is required for the tasks, as well as selective mechanisms to drive the search appropriately.

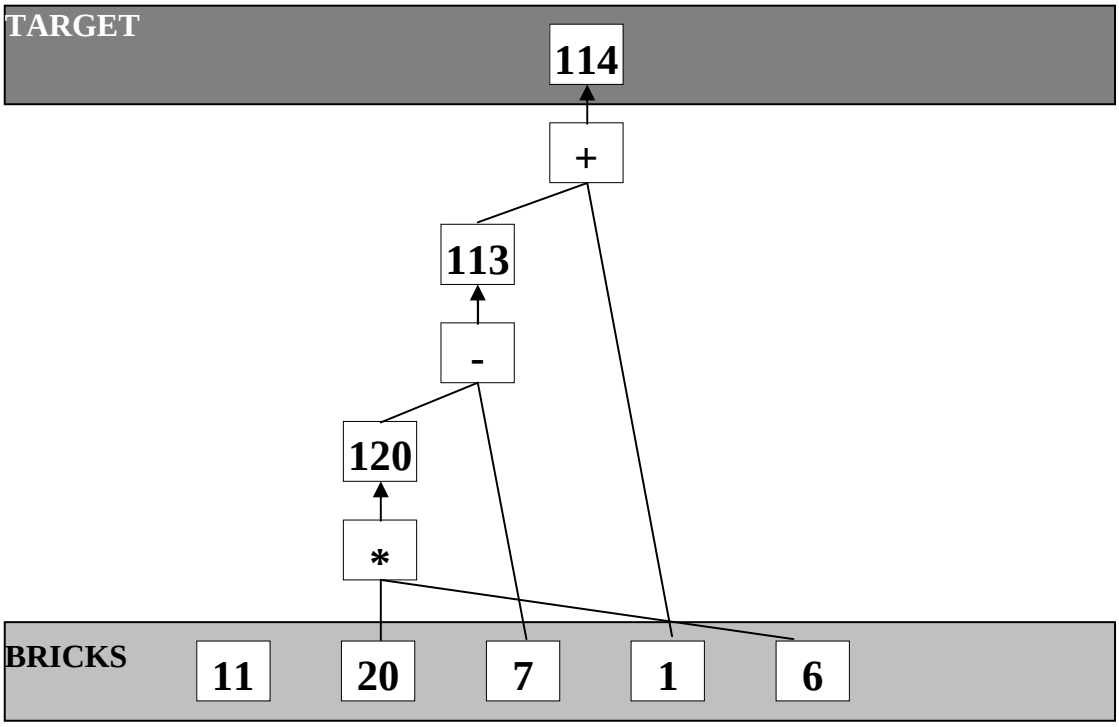


Figure 6.1 A hierarchical tree representing the solution 20*6-7+1=114 to a given puzzle. Pairs of numbers are combined through numerical operations to create new numbers that can be further combined.

6.1.3 Ability 5: The ability to exhibit rational, goal-directed behaviour

In the Numble domain, many combinations of bricks and operators do not represent valid solutions. For example, in figure 6.1, an exhaustive search would consider such possibilities as (20*11)-(7+1)*6, which leads to a number nowhere close to the required target (114). Such an approach is neither scalable, nor representative of human cognition. In solving such problems, humans generally use heuristics such as means-end analysis, where they choose operators that will generate a number close to the target (such as creating 120 from 20 and 6). From there, subgoals can be formed to direct attention towards filling in the missing steps (i.e. creating a subgoal of 6, which is the difference between the target and 120).

In addition to modelling the subgoaling behaviour of humans, the original Numbo program also captured the differences in salience of certain numerical operators over others. For example, given the bricks 20, 19, 6, 5 and 1 and a target of 114, human subject are more likely to produce the solution (20*6)-(5+1) over (19*6), displaying a

preference for 20*6 than 19*6. According to Hofstadter, this bias represents a dichotomy in the processes required, with the result of multiplying 20 and 6 being stored in declarative memory due to its common usage. Multiplying 19 and 6 by contrast, is an infrequent event, and requires procedural knowledge about how to multiply such numbers (though it can be reasoned that the result is close to 120 due to the similarity with the previous fact).

6.1.4 Ability 6: The ability to perform self-watching and mental regulation

In solving Numble problems, heuristics such as means-end analysis are often useful in directing the search. However, as will be discussed in the example runs given in this chapter, in many cases this bias can result in a dead-end solution. In such cases, self-watching and mental regulation is required to backtrack out of dead-ends and to search for new alternatives.

6.2 The FAE Implementation of Numbo

The FAE implementation of Numbo utilises representations and transformations based on the original Numbo model (Hofstadter & FARG, 1995). Firstly, a wide range of instances of salient addition and multiplication facts are stored in FAE's long-term memory in a semantic network similar to Numbo's Pnet (see 6.3.1). Secondly, FAE's working memory holds a hierarchical representation of the problem and currently explored solution, utilising similar representations to Numbo's workspace (see figure 6.3.2). Finally, all transformations in the workspace are carried out by a small number of productions that play a similar role to Numbo's Codelets (see figure 6.3.3). The main constituents of the FAE implementation of Numbo are described in the following sections.

6.2.1 FAE's Long Term Memory

In following the original Numbo implementation, FAE's long-term memory holds a wide range of instances of addition and multiplication facts that may be useful in solving the problems in its domain. In this regard, there are three main types of concepts in memory: Numbers (salient numbers between 1 and 150), and *addition_instances* and *multiplication_instances*. Addition and substraction instance nodes are connected to numbers through three types of links, *op1*, *op2* and *result* representing the two initial numbers that are combined and the subsequent result. Example facts stored in the network include facts that 1+2=3, 2+3=5, 2*2=4 and 3*4=12 (see figure 6.2). The current implementation has a collection of 45 salient numbers, 30 addition facts, and 32 multiplication facts stored explicitly in the network, although this number can be readily extended.

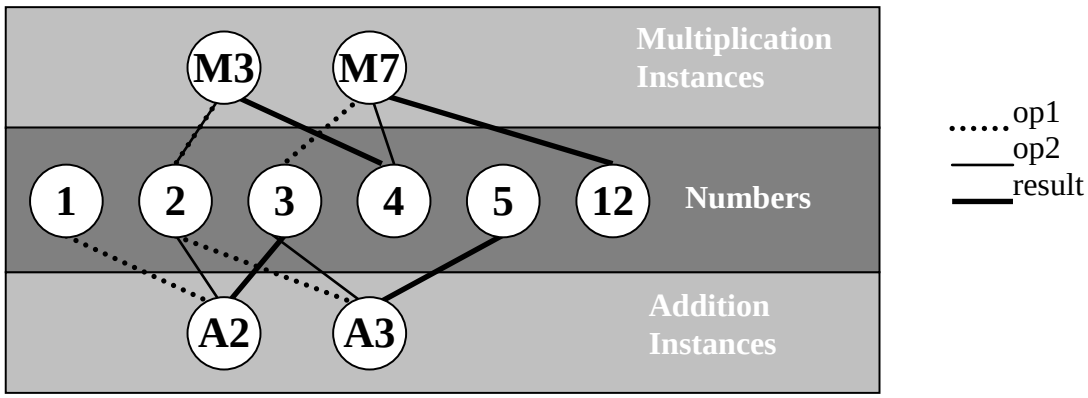


Figure 6.2 A subsection of FAE's long-term memory. The memory contains a set of numbers, as well as addition and multiplication facts that may be relevant to the problem solving domain.

As stated in previous chapters, concepts in FAE are represented in terms of distributed vectors. In the current case, *multiplication instances*, *addition instances* and the various link types are represented using a local coding scheme in which a single bit is on for each different instance. Numbers are also represented using a local coding scheme, with each bit corresponding to a different integer value.

The heuristics used during search require a measure of similarity in representing numbers. For example, if trying to build a target of 117, and the bricks 6 and 20 are

available, it should be noted that their product (120) is close to the required target. To deal with such cases in FAE, productions require a "close to" operator that is specific to numbers. This similarity value follows a gaussian distribution, normalised to the larger value, so that if they are equal, a similarity value of 1.0 is returned, with a lesser value otherwise. The similarity value *s* between two numbers *n1* and *n2* is defined as follows:

$$s = e^{-cn^2} \tag{1}$$

where *c* is a constant (50.0), and *n* is the normalised difference between the two numbers:

$$n = \frac{\max(n1, n2) - \min(n1, n2)}{\max(n1, n2)} \tag{2}$$

Numbers in the current domain are always positive non-zero integers, making the value of similarity always calculable.

6.2.2 Working Memory

Initially the workspace contains a collection of 5 bricks and a single target. Each of these entities is represented by a separate object with two properties: *Object Type* (either *brick, block* or *target*) and *Number Value* (an integer value). In building a solution, two main objects can be built: *blocks* (that are the result of an arithmetic operators), and *subgoals* (that represent the difference between the target and similar bricks or blocks). Blocks are linked to the two numbers that have been combined to form the new value, and have the additional property *Operation* that expresses how the new value was formed (*add, subtract,* or *multiply*) (see figure 6.3a). Subgoals, in contrast, create two new blocks, one representing the new target value, and the other representing how this target was derived (an addition or subtraction between the previous target and the closest matching object) (see figure 6.3b). In both cases, when these structures are formed in the Workspace, the properties of the objects upon which they are formed become suppressed, and invisible to Productions (as only each object can be used at most once). When the current target is found to be equal to an existing brick or block, a link between them is built, indicating that a solution has been found (see figure 6.3c).

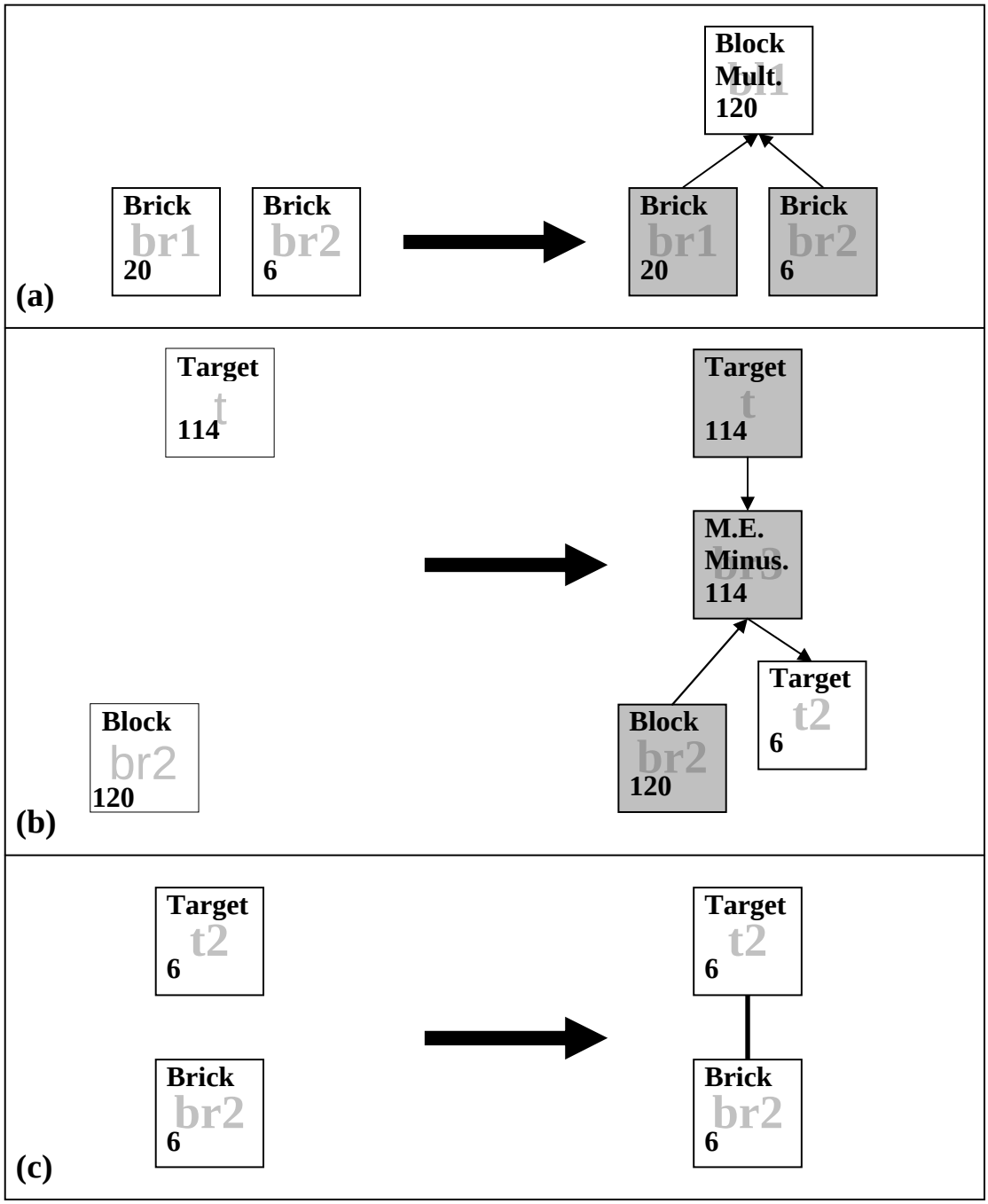


Figure 6.3 The structures that can be formed in the FAE implementation of Numbo. (a) Given two free bricks or blocks, a new block can be formed encoding the addition, subtraction or multiplication of the two values. (b) Given a brick or block close to the target, a new subgoal can be formed (i.e. a new target) representing the difference between the values. (c) if there exists a free brick or block of equal value to the target, the two can be connected, indicating that a solution has been found.

6.2.3 Productions

Productions in FAE take the role of the Codelets found in the FARG models. As discussed previously, all productions fire in parallel, each exerting variable degrees of pressure upon the Workspace depending upon how well the preconditions are matched. In the FAE implementation, in total, nine Productions are used, two for combining bricks through multiplication, two for addition, two for subtraction, two for creating subgoals and a single Production for termination (if there exists a free brick or block with a value equal to the current target).

In performing arithmetic operations, two separate productions are used for either addition, substraction or multiplication to emulate the dichotomy between declarative and procedural tasks captured by the original Numbo model. That is, for example, if there exists two free bricks (such as 6 and 20), which can be combined using a fact stored in long-term memory (i.e. 6*20=120), this process should be relatively fast, being captured by the declarative memory production. By contrast, if there exist bricks that are close to, but not identical to, values that are included in a long-term memory chunk (such as 6*19), it can be assumed that the combination of these values will be close to the stored chunk value (6*20=120). However, in this case, multiplying 6 by 19, uses a slower mental mechanism (procedural memory for performing the multiplication). This dichotomy is represented in the FAE model by using a separate but slower production for dealing with such uncommon combinations, mimicking the preferences of human subjects to choose operators that use well-learned facts.

Block Building Productions

Block-Building Productions utilize a form of means-end analysis to give preference to the formation of structures that are close to the current target. That is, operations are not randomly chosen, but are applied in a rational manner that will bring the system closer to its current goal. Such Productions are reliant on the facts stored in long-term memory. As stated, there are two such Productions for multiplication, subtraction and addition, the first utilising declarative knowledge (explicitly retrieving facts from

long-term memory), and the second representing procedural knowledge (e.g., the multiplication of two numbers).

Both declarative and procedural productions have similar forms (shown diagrammatically in figure 6.4). For the declarative productions, the preconditions inspect two bricks or blocks, performing an explicit retrieval to calculate their combined values. If the retrieved result is close to the current target, the production is fired. The degree of firing however, is a function of how close the resulting number is to the current target, favouring the formation of closer results. Procedural productions by contrast, retrieve chunks that have similar, but not necessarily identical values to the chosen bricks or blocks (such as retrieving 20*5=100, given the values 19 and 5). If the retrieved chunk has an arithmetic combination result that is close to the target, the production again will fire to a variable degree. However, as the combination of the chosen workspace numbers is not explicitly known, this value must be mentally calculated (e.g., using procedural knowledge to multiply 19 and 5). In the current model, this process is carried out within the procedural production (e.g. using program code to multiply the two numbers). However, these productions are given a lower precondition match to their declarative process counterparts, thus being performed at a much slower rate and being less preferred over declarative productions.

Apart from specifying what structures should be formed in the Workspace, each production also specifies what structures need to be destroyed for the proposed structure to form. That is, incompatibilities are specified within the productions themselves. In the Numble case, as each number can only be used once, for a new block to be created, any other structure that incorporates either of the numbers that it combines needs to be destroyed.

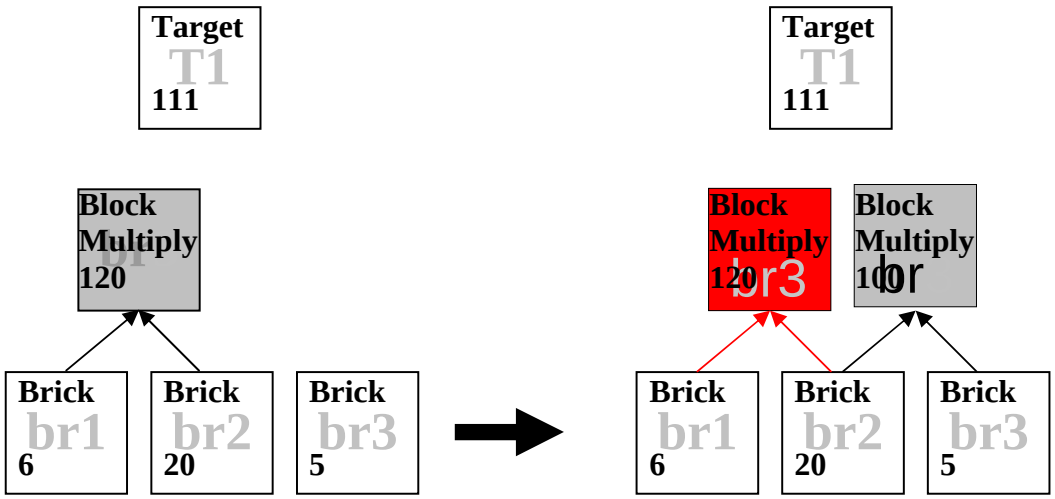


Figure 6.4. Building of a multiplication chunk. Given that the bricks 20, and 5 are present with the target 111, the chunk 5*20=100 will become highly active in LTM due to spreading activation. As the target value is close to the result of the multiplication, the preconditions of the multiplication production will be satisfied, with a new multiplication block being proposed. In the above case, the new block will be incompatible with the existing structure as they both use the 20 brick.

Subgoaling Productions

Subgoaling Productions are used to determine the difference between the current target and the closest matching brick or block, with a subgoal of the difference between them being created when the values are close (roughly within 20% of each other). Subgoaling Productions try to form two explicit objects in the workspace: a new target value, equalling the difference in values between the target and the closest match, and an operator object, determining if this difference is either an addition or subtraction depending on whether the closest value is less than or greater than the current target (see figure 6.5).

Subgoaling productions are important in triggering new block building productions to bridge the current gap. For example, if the current target is 114, the closest value is 120, and there exist two free bricks, 2 and 3, the two free bricks will not be multiplied into the bridging value of 6. This is because the multiplication of 2 and 3 is **not** close to the target value, which is the first precondition of the block building codelets. However, a subgoaling production calculates the difference between the target (114)

and the closest value (120), and builds this as a new target (6). The block building production can then become activated and bridge the current gap.

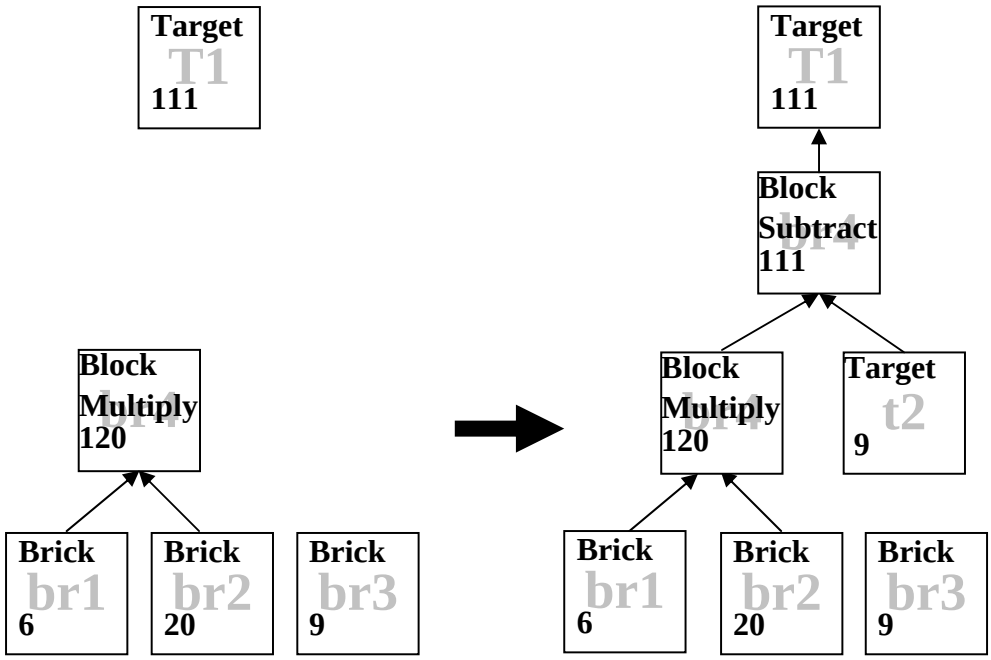


Figure 6.5. The results of a subgoaling production. Given the current target of 111, and a similar matching value of 120 in the Workspace, the difference between these two values is explicitly calculated, with a new target being built denoting the difference, as well as a block that specifies the operator required.

Termination Production

The last production utilised in this implementation of Numbo is a termination production that determines when a solution has been found. This production simply identifies when a "free" brick or block equals the current target value (e.g. the current subgoal). When this occurs, the two objects are linked, and the system halts (see figure 6.6).

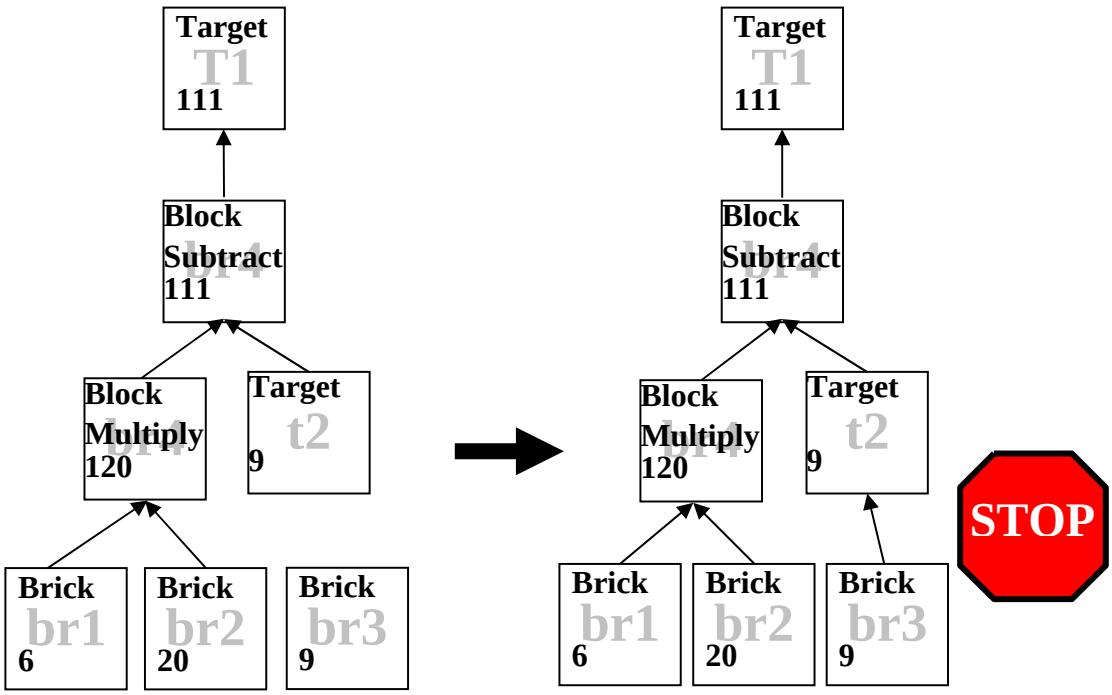


Figure 6.6 Program termination. When an unbounded brick or block is found with the identical value of the current target, the two structures are linked and the system halts. The answer to the problem is expressed by the resulting tree structure: (6*20)-9=111

Dealing with Structural Incompatabilities

The ability to resolve conflicts is central to the Numble domain as each brick can be combined with others in a number of different ways, with the limitation that only one such combination can be used (i.e. each brick can be used at most once). Such incompatibilities are defined within each production (as this is task dependent), allowing for each production instance to identify which structures in the workspace it is competing with. As explained in chapter four, such conflicts are resolved locally in FAE through the use of its attractor network. As with typical constraint satisfaction networks, incompatible structures are linked with inhibitory connections, resulting in the suppression of alternatives as the prominent solution gains activation. In Numbo, this inhibition prevents the reuse of bricks, allowing only a single operator to be explored at a time.

Backtracking through Dead-ends

One of the problems with means-end analysis, especially in the game of Numble, is that sometimes operations that seem to bring the system closer to a solution can lead to an impasse. For example, given the target of 119 and bricks of 20, 6, 19, 2 and 5, multiplying 6 and 20, although bringing the state close to 119 with a single operation, will lead to a dead-end, as there is no way to bridge the gap of 1 by combining the free bricks. In order to deal with impasses, FAE requires a mechanism for backtracking that destroys structures that do not lead to a solution, allowing the system to explore the alternative paths. In dealing with such impasses, it is also beneficial for a system to be able to remember the path, avoiding the same pitfall during subsequent explorations. FAE deals with this issue by having a form of episodic memory, in which sequences of events are stored that can be avoided in future if they have not proven fruitful.

In FAE, the strength of a trace of a structure in working memory is a function of the duration in which it is "conscious" (i.e. above threshold). In the Numbo implementation, such "conscious" structures are bricks and blocks that have not been chunked into higher-order structures. In the formation of a solution, the longer a block is free (i.e. not combined further with operators), the more likely it is that the structure is not useful. In FAE, objects gain episodic strength over time, with the amount of suppression placed on the formation of such structures being related to this strength. Thus, if an object is formed in the Workspace, and no other objects are formed using it to satisfy a precondition, the episodic trace will increase, resulting in increases in inhibition, with the structure eventually being destroyed. This process allows FAE to backtrack through dead-end solutions and avoid the state on further explorations.

6.3 Example Runs

The following sections describe two example puzzles explored by the FAE implementation of Numbo that demonstrate the main aspects of problem solving

utilised by the program: representation formation, conflict resolution, means-end analysis and backtracking.

6.3.1 Puzzle #1

In puzzle #1 described there are 5 bricks having the values of 11, 20, 7, 1 and 6, and a target of value 114. This puzzle is interesting as humans generally form the solution 20*6-7+1, rather than the more optimal solution of (20-1)*6 (Hofstadter and FARG, 1995). The following simulation replicates this result.

Solution Formation

Although there are many possible operators that can be used from the initial state, block building productions try to form numbers that are close to the current target. Thus the multiplication of 6 and 20 is performed, creating a block with the value of 120 (figure 6.7a). When this block is formed, it acts as a chunk by reducing the attention paid to the underlying two bricks, preventing them from being considered by other productions (figure 6.7b).

As the new block (120) is close to the current target (114), a subgoaling production will fire that creates a new target that denotes the difference between these numbers. In this case, a new target (6) is created, with the original target being suppressed. Furthermore, an operation block is also added to the workspace that specifies the operator that is required to reach the new target from the old. In this case, the old target (114) can be reached through the subtraction of the new target (6) with the "120" block (see figure 6.7c).

In its current state, the workspace now contains the salient numbers 11, 7 and 1 as bricks and blocks, and the target of 6. Now that the current target can be achieved utilising these values, a block building production is fired, subtracting 1 from 7 to create the new block value of 6 (figure 6.7d). As this new block is equal to the current target, this correspondence is formed, with the system halting with the solution (20*6)-(7-1) (the solution preferred by humans).

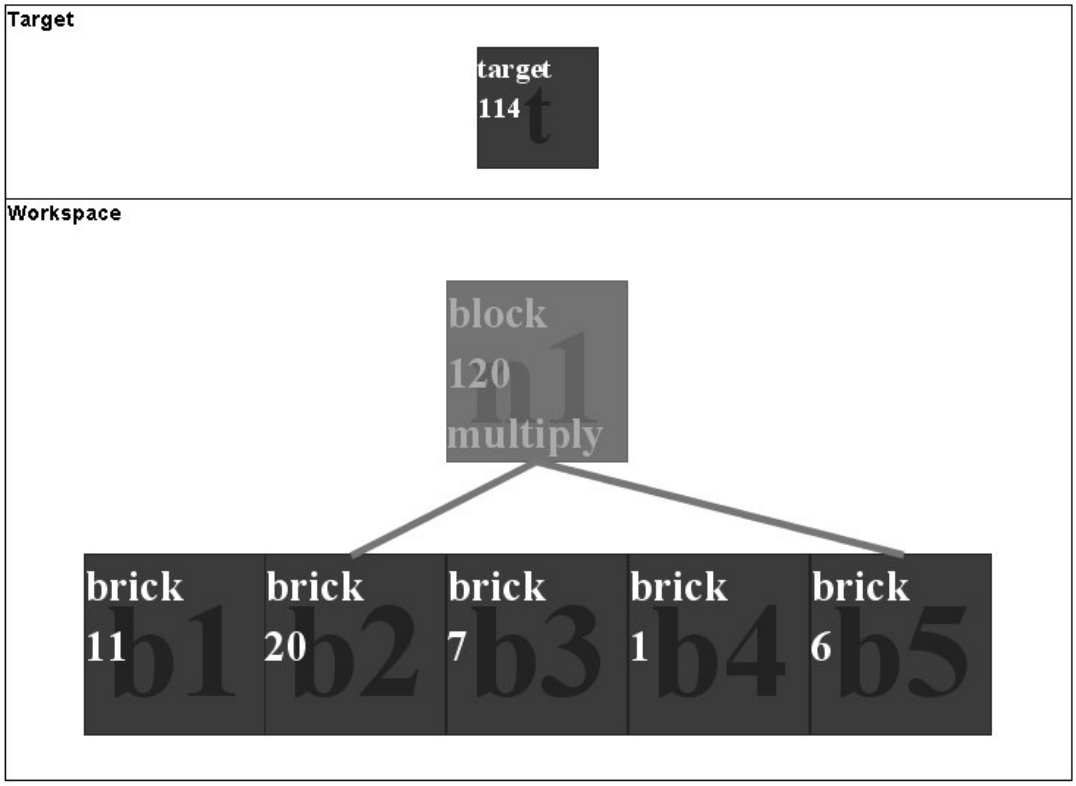


Figure 6.7a Given problem #1, FAE starts by multiplying 20*6 in order to form a block close to the target.

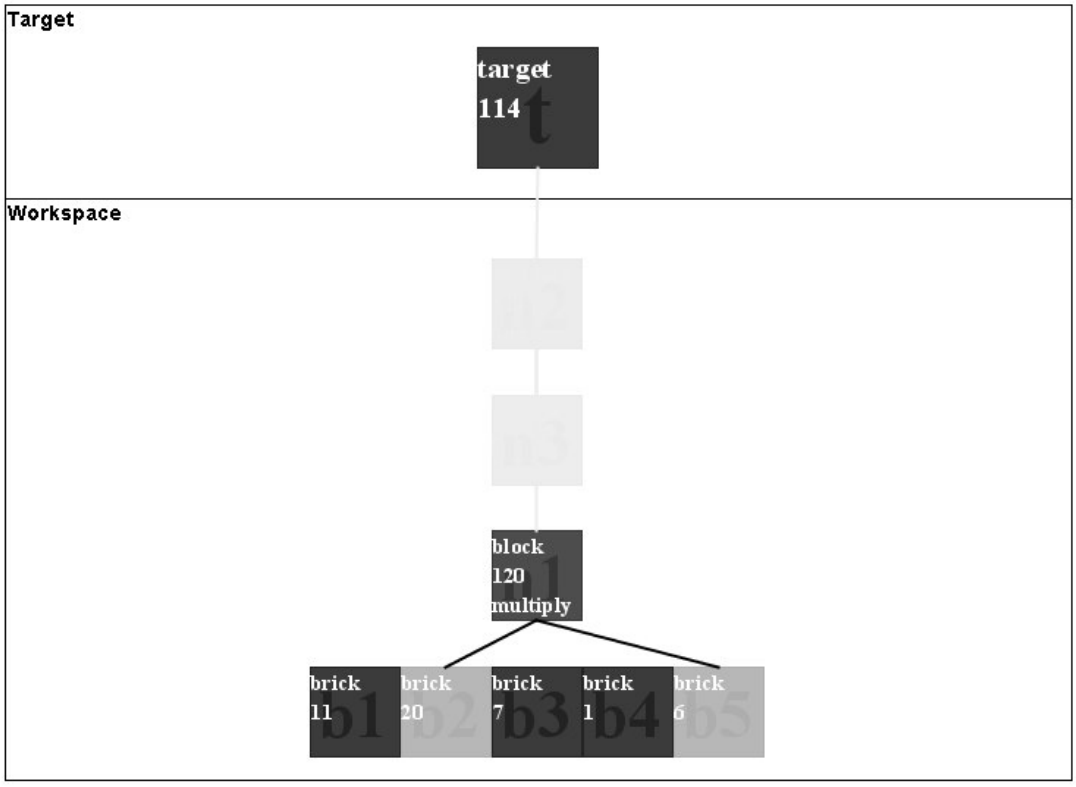


Figure 6.7b When the new block becomes fully formed, the underlying bricks are suppressed, focussing attention on the available numbers that can be used.

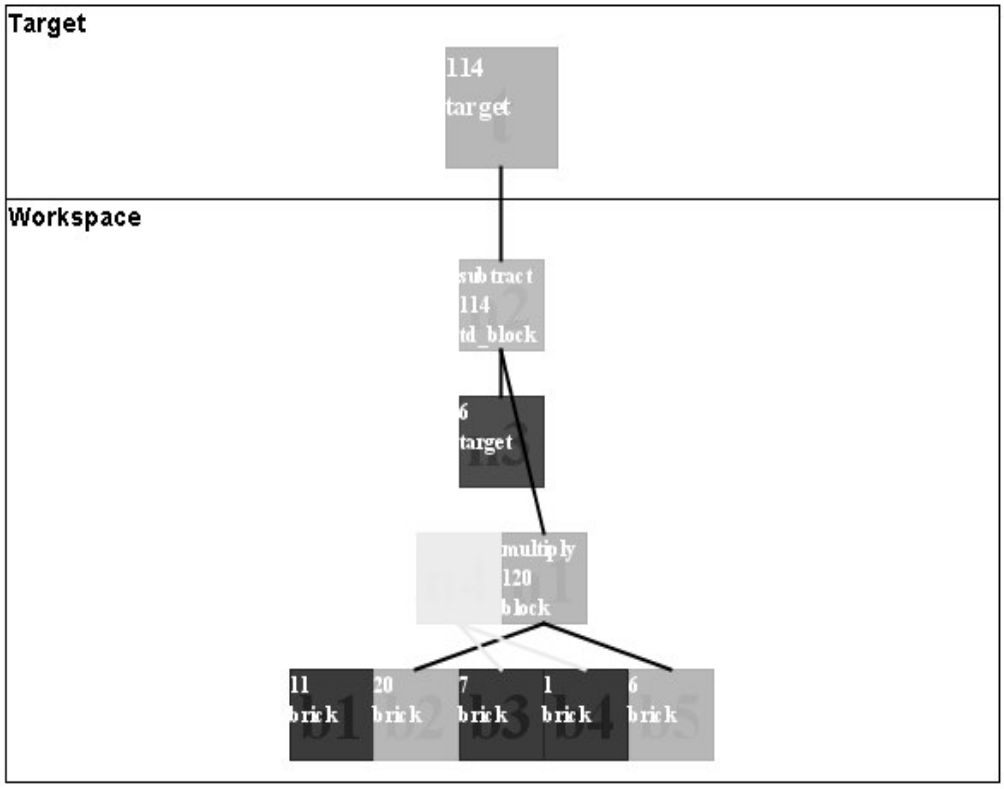


Figure 6.7c As the new block is close to a target, a subgoal is created, to see if the difference can be formed.

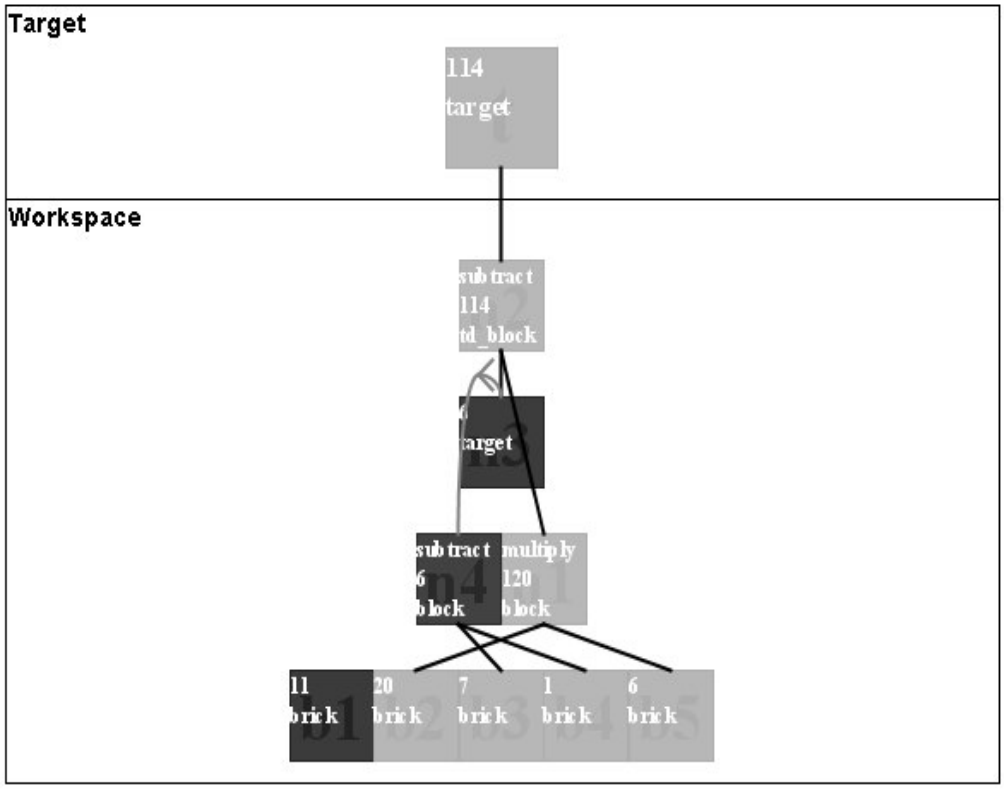


Figure 6.7d As the subtraction of 1 from 7 creates a block close to (i.e. equal to) the target, this action is formed. As the resulting block equals the target, this correspondence is formed, with the program terminating.

6.3.2 Puzzle #2

The following puzzle is interesting in that backtracking is generally required due to the heuristics driving the system towards a dead-end solution.

Puzzle#2: Target: 102
Bricks: 20, 19, 5, 3, 4

In this problem, if following a means-end strategy, the first operation that should be selected is 20*5, as it results in a value (100) quite close to the target. However, in this case, the remaining bricks cannot be combined to bridge the gap of 2 that remains. Instead, the two possible solutions include 19*5+4+3, and 20*4+19+3. The following sections detail a run of FAE that results in the former solution, although both are possible.

Initial Exploration

As both 5*19 and 5*20 create results close to the target (102), two production instances become triggered: a declarative production trying to build the block "100" from multiplying 5 and 20, and a procedural production, that tries to multiply 5*19. Although these structures are incompatible (because they both use the brick 5), they both "bubble up" into consciousness, fighting for supremacy (figure 6.8a).

Conflict Resolution & Subgaoling

Although the structures representing the multiplication of 100 and 95 arise in parallel, the structure formed by the declarative production will gain in strength at a faster rate. Because these two structures are incompatible, they inhibit the formation of each other, with a strength proportional to their current strength. Thus, as one structure dominates at a high strength level, the other will be suppressed (figure 6.8b). In this case, due to the differences in saliency between the two proposed structures, the fact built from the declarative production will dominate (i.e. 5*20). Once fully built, the "100" block acts as a chunk, suppressing the attention given to the constituent bricks 5 and 20 and preventing the procedural production from continuing firing.

Once the block “100” has been formed, a subgoaling production will fire, due to the close proximity of this block to the current target (102). This production will result in a new target being formed (2), with the suppression to the “100” block and the main target (figure 6.8c).

Backtracking

Given the new target of “2”, and the available bricks 19, 3 and 4, there is no combination of the free bricks that can form the target. Thus, the system is at an impasse at this point. In this state, the strength in episodic memory of the newly formed structures slowly increases over time. As no further structure is built upon the new target, the episodic trace reaches a threshold and the subgoal is inhibited, freeing up its underlying constituents (figure 6.8d).

Once the system has backtracked out of the first proposed subgoal leaving the 100 block free once again (figure 6.8e), the episodic strength of this structure starts to climb. Again, as no other combinations are attempted that include this structure, the 100 block is also destroyed, resulting in the initial problem state.

Exploration of Alternative Solutions

The block representing the multiplication of 5 and 20 blocked the other possibility of multiplying the 19 and 5. Once the 100 block is destroyed, this alternative can be explored. Again, due to the similarity of the newly formed block (95) with the target (102), a subgoaling production will fire, creating the new subgoal of 7 (figure 6.8f).

Solution Formation

As the addition of 3 and 4 is close to (actually equal to) the current target, the block 7 is formed. As there is now a free block that is equal to the current target, the termination production will be triggered, building a bond between these objects, and halting the system (figure 6.8g), resulting in the following solution:

102 = (19*5)+(3+4)

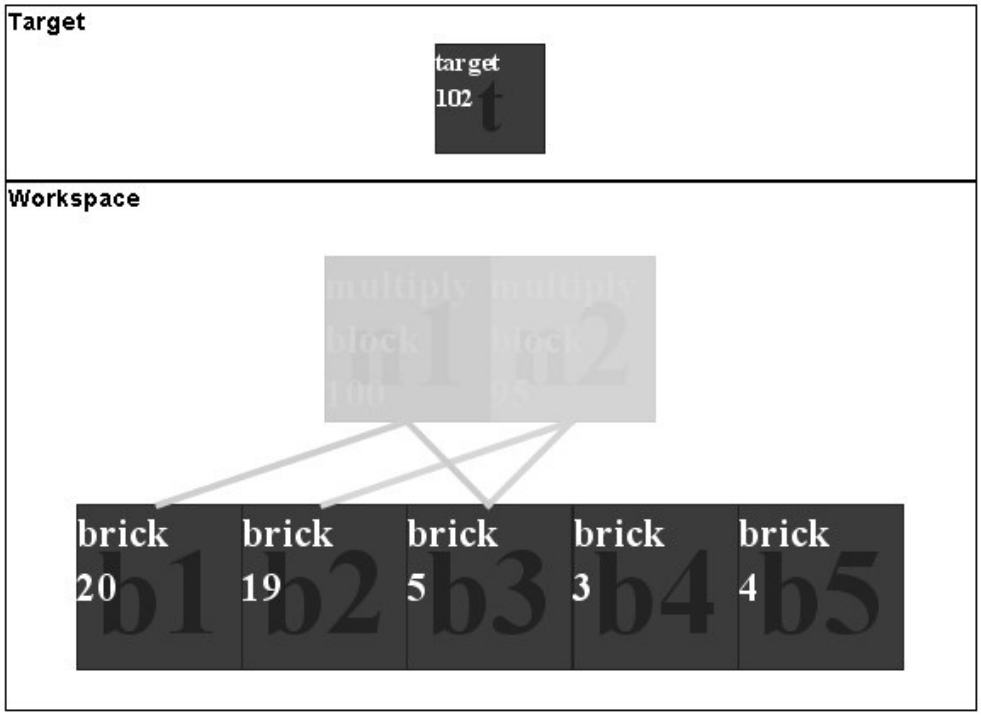


Figure 6.8a Given problem #2, there are initially two competing operations that can be applied to create a block close to the target: the multiplication of 20 and 5, and the multiplication of 19 and 5.

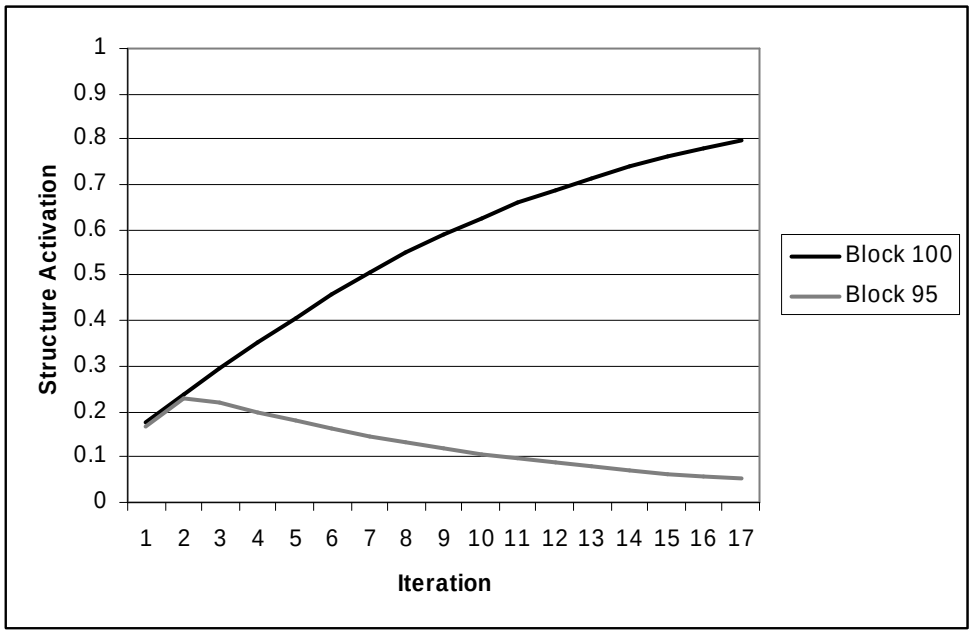


Figure 6.8b As the two solutions in Figure 6.8a are mutually exclusive (as they both use the 5 brick), the structures are connected through negative weights. As the multiplication of 20 by 5 is more salient than 19 by 5, this solution becomes active over time, suppressing the alternative.

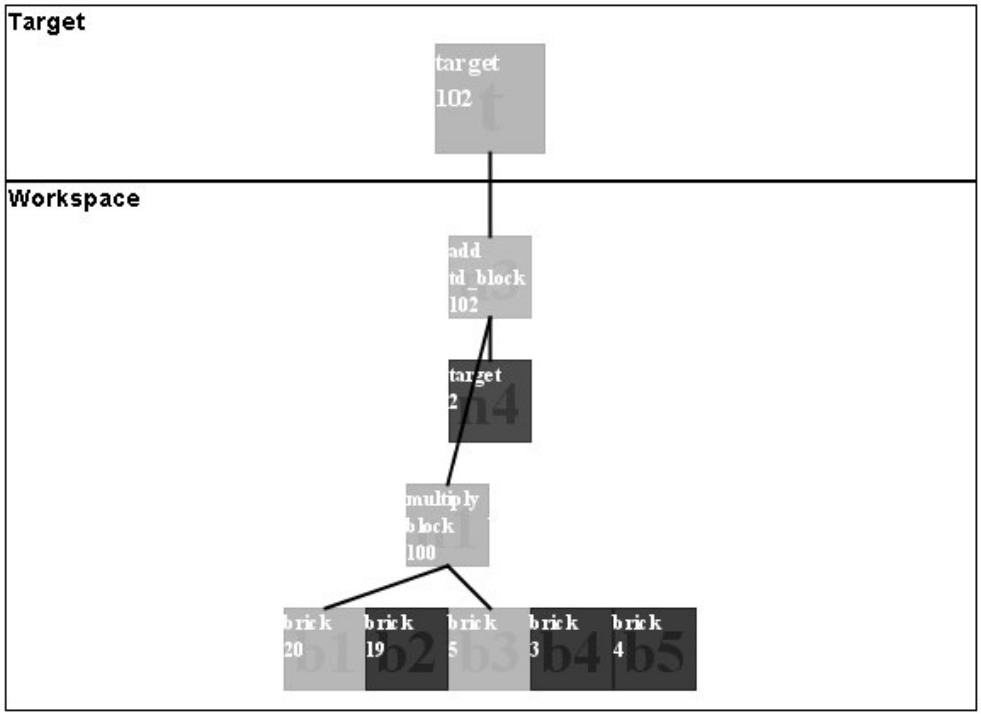


Figure 6.8c Due to the similarity between the current target and the newly created 100 block, the difference between these two values is explicitly calculated, creating a new subgoal of 2.

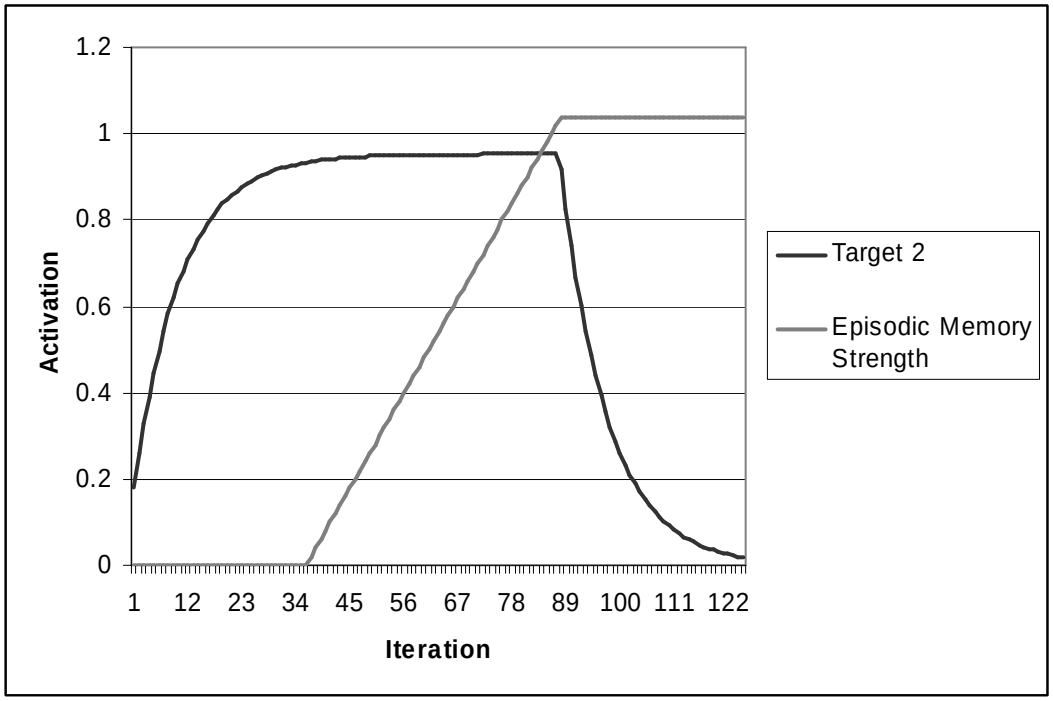


Figure 6.8d Once the activation of the new structure exceeds a threshold value, an episodic trace of the current workspace state (i.e. the set of active numbers) is created, gaining activation over time. Once a threshold episodic memory strength is exceeded (indicating that an impasse has been reached), the newly formed structure is inhibited.

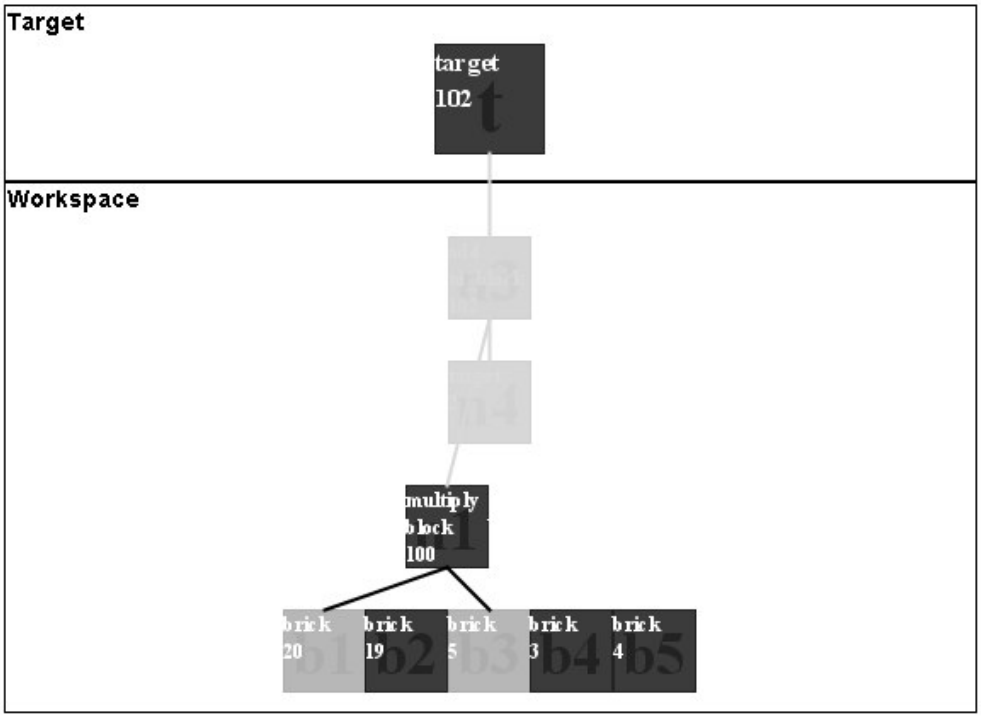


Figure 6.8e With the destruction of the new subgoal through inhibition from episodic memory , the system backtracks to the previous state.

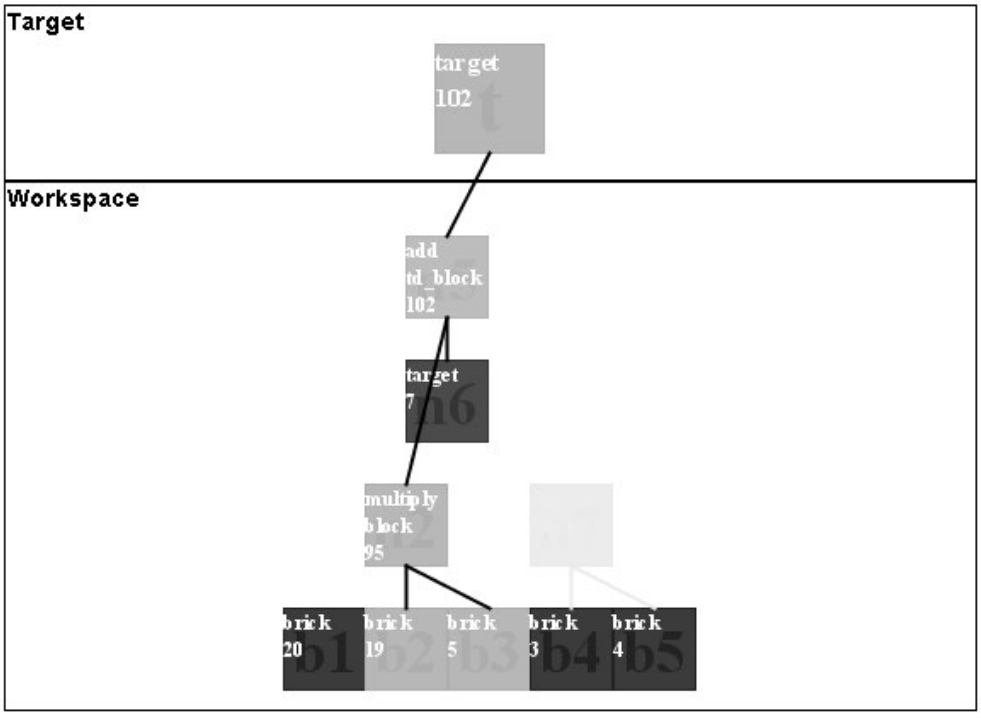


Figure 6.8f Over time, the exploration of 5*20 is also abandoned, allowing the multiplication of 19*5 to be explored, resulting in a difference of 7 from the target.

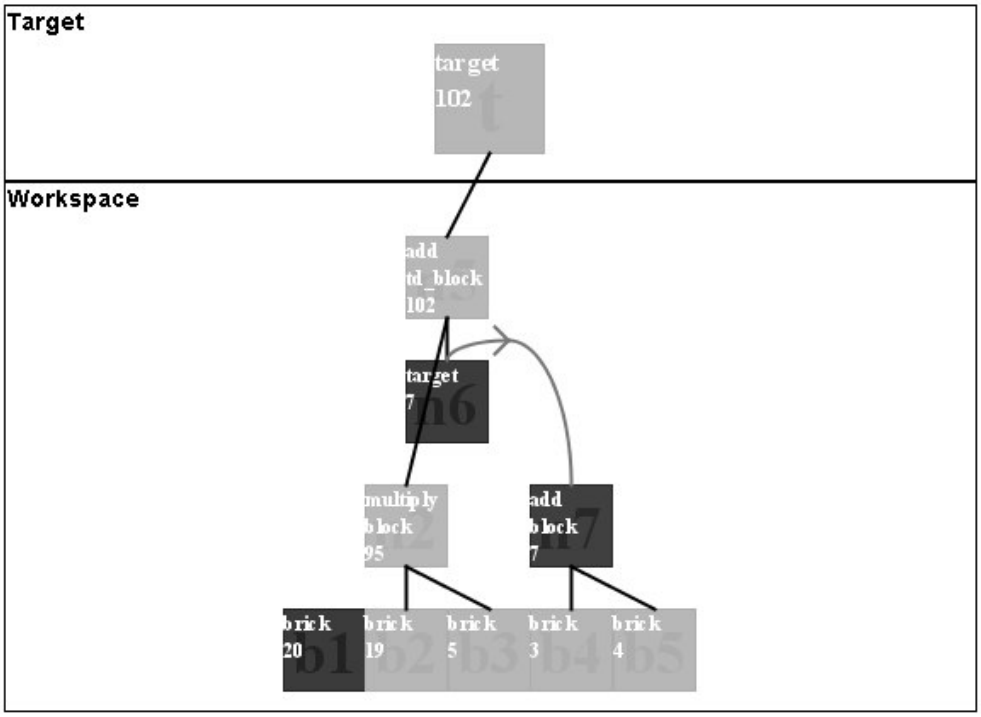


Figure 6.8g The new subgoal can be achieved through the addition of 3 and 4, resulting in a valid solution.

6.4 Discussion

In this chapter, an implementation of FAE that plays the game of Numble was described. The Numble domain is interesting as, to implement a working model, many questions central to high-level perception need to be addressed. Firstly, what is the nature of representations in working memory and long-term memory? Secondly, how can a large search space be explored without a combinatorial explosion of possibilities? Thirdly, how can a large amount of information about the world be represented and efficiently utilised? And fourthly, what is the nature and heuristics of perceptual chunking that allows some percepts to be interpreted as a higher order group with their own set of unique features?

At a general level, the answers that FAE provides for the above questions are consistent with the previous models implemented by FARG. Firstly, in Working Memory, percepts are represented by a set of objects, properties of objects, and relationships between the objects, all being represented by a set of predefined

symbols. This approach is also consistent with the working memories of SOAR and ACT-R. FAE and the Fluid Analogy (FA) models differ from these two models however, by assigning activation levels to each structure to allow for a degree of "subconscious" exploration of the space.

In terms of utilising heuristics to explore and reduce the search space within this domain, the four previously mentioned models differ substantially. Both FAE and the FA models (such as Numbo) use a stochastic search, in which only one interpretation is represented in "consciousness" (i.e. above threshold) but with many "subconscious" explorations close to the current interpretation being carried out in parallel. In these models, all incompatibilities are resolved at the subsymbolic level, with "subconscious" explorations having to destroy their "conscious" competitors in order to become perceived. In all of the FA models apart from Metacat, the search is stochastic, requiring a form of simulated annealing to backtrack out of suboptimal solutions. Metacat also incorporates this form of annealing, but includes an episodic memory that reduces the likelihood that poor solutions will be re-explored. In these models, the temperature value that controls the degree of randomness is a function of the quality of the structure that is built in the workspace. That is, good solutions should have a low temperature value that equates to a low degree of randomness that is required to avoid a stochastic breaking free from the solution. Thus, in order to apply the general FA method to a new task domain, a function for determining appropriate temperature values and levels of randomness from the built structures needs to be defined. Deriving such a function is not necessarily an easy task, as each different task requires a different algorithm. For example, in Numbo, the temperature is related to the similarity between the target and closest value, whereas in Copycat, the Temperature value is a complex integration of all the built bonds and correspondences between objects. FAE, by contrast, does not utilise simulated annealing, rather relying on its episodic memory to backtrack through poor solutions. This mechanism provides a more generic solution, as no algorithm for calculating temperature and equating this value with appropriate levels of randomness is required. Furthermore, unlike models such as Copycat that will re-explore poor solutions many times, FAE's episodic memory stores a trace of all possibilities explored so far, making it easy for the system to avoid retracing suboptimal paths.

The FAE implementation of Numbo uses a form of means-end analysis to bias the selection of operators towards those that take the system closer to the solution (the target value). That is, at each stage there may be several possible incompatible operators that can be selected, with only one being resolved (through competition at the subsymbolic level). Superficially it can be said that ACT-R works in a similar manner, with concepts in long-term memory taking variable degrees of activation, with only one being probabilistically chosen during a memory retrieval. That is, it is foreseeable that a probabilistic selection of operators that is consistent with the means-end analysis that is used by FAE may be implemented in ACT-R. However, as seen in the second Numble example explored in this chapter, such a means-end analysis by itself can often lead down the garden path. As ACT-R does not have a mechanism for backtracking out of such solutions, or keeping track of what solutions have already been explored, it is unrealistic to assume that ACT-R in its current form would be capable of the same problem solving capabilities as FAE.

SOAR, unlike ACT-R, may contain the requisite mechanisms in order function effectively in domains such as Numble. In SOAR, during the decision cycle, many productions fire in parallel to accumulate evidence in order to resolve the selection of the next operator. There are two mechanisms that could be employed by SOAR in order to choose the appropriate operator. Firstly, there could be information stored, that would lead to a means-end analysis, preferring the selection of operators that bring the system closer to the target. However, by itself, like ACT-R, this approach would often lead the system into a dead end that the system could not escape from. Unlike ACT-R however, SOAR does have a search mechanism that allows it to escape from such impasses: subgoaling. Whenever there are no operators that can be used from the current state, or there are several operators that the system cannot decide between, a new state is created to draw in information that may resolve the conflict. Such states can themselves create substates, implementing a goal hierarchy that can be used for a lookahead search. It is foreseeable that in the domain of Numble, using such a lookahead procedure coupled with the heuristic selection of operators, a solution could be effectively found. However, this is not surprising, as SOAR was developed specifically to handle such symbolic searches through large problem spaces.

6.5 Summary

This chapter describes an implementation of the game of Numble using the Fluid Analogies Engine. Numble is an important domain to model as it is representative of a large class of tasks in which a set goal needs to be achieved through the selection of nested operators. In order to perform this task effectively, four of the six behavioural properties outlined in chapter one are required: the ability to create and manipulate complex hierarchical symbolic structures, the ability to use selective attention in lieu of unlimited cognitive resources, the ability to use rational behaviour to explore many alternatives without a combinatorial explosion, and the ability to self-watch in order to avoid re-exploring poor solutions. At the gross level, FAE emulates the behaviour of the original Numbo implementation (Hofstadter & FARG, 1995), utilising the same kinds of representations and stored knowledge. FAE differs from Numbo however, with the inclusion of an episodic memory that affords a form of self-watching that regulates the resources (though inhibition) to backtrack out of impasse situations and avoid previously explored avenues.

Chapter 7
Modelling Creativity

Creativity can be viewed as the act of coming up with something that is both novel and intelligible (e.g., Boden, 1994). In "coming up" with such instances of novelty, high-order rules or regularities that exist within the specified domain can often be exploited. For example, in the composition of music there are definite statistical regularities that exist, such as typical chord progressions and the bias to select notes (that occur on the beat) from those in the tonic triad. In computer models of creativity, such regularities have often been turned into grammars that allow for the construction of novel entities. Such grammar-based systems have been used in numerous domains including the design of vehicles and houses and in the generation of stylistic paintings (e.g., Pugliese & Cagan, 2002; Stiny & Mitchell, 1978; Koning & Eizenberg, 1981; Flemming 1987; Kirsch & Kirsch, 1986).

This chapter explores the ability of FAE to exhibit creativity using the domain of anagram-formation as a test-bed. This process exemplifies creativity in the sense that, during solution formation, humans often generate word-like candidates (words that conform to the general statistics of word formation), stopping only when a valid word has been generated (Hofstadter & FARG, 1995). Unlike in grammar based systems in which a solution is formed from the top down, solving anagrams requires the multi-level cleaving, splicing, regrouping, reordering and rearranging of elements and groupings already perceived, thus requiring an integration between bottom-up and top-down processes. According to Hofstadter, "such operations permeate the process of creation, whether it is composition of music, art, or literature, or the invention or discovery of new ideas in science" (1995, p100). Thus, although the game of Jumble

seems relatively straightforward, it requires the use of many processes believed to be central to creativity that are not explored by simple grammar based systems.

7.1 Cognitive Processes Required

According to many researchers, the ability to create something novel does not necessarily require a separate "creativity module", but can arise through the same set of mechanisms underlying such skills as problem solving and flexible perception (Weisberg, 1988; Hofstadter & FARG, 1995). For example, as with planning, anagram formation relies on the ability to create and manipulate complex hierarchical structures, the ability to focus attention on a particular region of the search space (in lieu of being able to explore all combinations in parallel), the ability to exhibit rational goal-directed behaviour, and the ability to perform self-watching and mental regulation. Thus, it is speculated that there are no extra components of FAE that are required in displaying creativity (such as forming anagrams) beyond those described in previous chapters.

7.2 The FAE Implementation of Jumble

The current model of anagram formation in FAE was inspired by the Jumbo project described in chapter two (Hofstadter, 1983). In Jumbo, anagrams were mostly solved through the bottom-up hierarchical chunking of letters into likely consonant and vowel clusters, clusters into syllables and syllables into words. Along the way, Jumbo was also able to recombine the sequences if they would result in a more statistically regular grouping (such as swapping consonant clusters between syllables). The FAE implementation similarly chunks sequences of letters into higher-order groups using a bottom-up approach, biased towards the formation of statistically regular sequences.

7.2.1 FAE's Long Term Memory

In the original Jumbo model, formation of letter sequences was probabilistically biased, utilising the information stored in the *p-net* (Jumbo's long term memory). This information specified the likelihood that letter strings (such as "s-t-r") would appear at the beginning, middle or end of a syllable. In the FAE implementation, similar facts are known to the system, being stored in FAE's semantic memory

network. In this network, two main facts about letter sequences are represented: the relative frequency of common vowel and consonant clusters (such as *s*-*t* or *o*-*u*), and the relative likelihood that the various letters or letter clusters will appear at either the beginning, middle or end of a syllable.

In FAE's semantic network, individual letters and common vowel and consonant clusters are represented by separate nodes. In this network, the cluster nodes (such as *s*-*h*) are connected to their constituent elements (i.e. the nodes *s* and *h*) with weighted links that reflect their relative frequency of occurrence (see figure 7.1). For example, the cluster "th", which is a frequently appearing combination, will be connected to "t" and "h" through strong weights, whereas the cluster "wn" will be connected to its elements through weaker links. Thus, through spreading activation, more frequent letter combinations will gain higher levels of activation when their constituent letters are present in working memory. When linked to the production system, this difference in activation level will result in frequently occurring sequences being more likely to be formed. In total, apart from the 26 letter nodes, there are 16 vowel clusters, and 37 consonant clusters that are stored in the network. The network also contains information about the relative likelihood that each sequence will appear at the beginning, middle or end of a syllable. This information is implemented through weighted connections to nodes representing *start*, *middle* and *end* (see figure 7.1).

Unlike in the original Numbo program in which relative weights were assigned by intuition, in the FAE implementation, weights were derived from the frequency of their occurrence in the Kucera and Francis word database (1967), being normalised to values between 0 and 1.

7.2.2 Working Memory

Initially FAE's working memory contains a set of objects corresponding to the individual letter instances in the given anagram. Each object is assigned two properties: an *object type* (which is *letter* in this case), and a *letter value* (in the range *a*-*z*). Through the actions of productions, such letters can be chunked into consonant and vowel clusters, suppressing the activation of the objects on which they are founded. Such suppression makes the objects invisible to the production system,

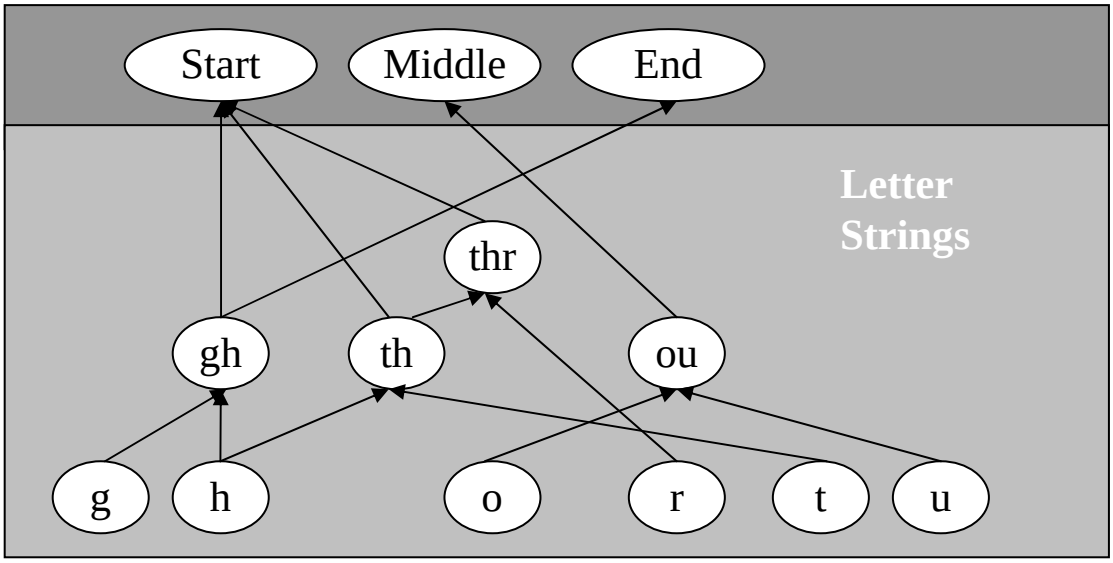


Figure 7.1 A subsection of FAE's long-term memory. The memory contains a set of letters, vowel clusters and consonant clusters that are connected through weighted links. For each letter string, the relative likelihood of occurrence at the beginning, middle or end of a syllable is refected by the weight of the connection to the corresponding position node

focussing the attention of the system on the manipulation of the higher-order units. Like letter objects, consonant clusters have two property values: the *object type* (*cluster* in this case), and *letter value* (equal to the letter cluster string). Based upon groups of letters and letter clusters, syllables and words can also form. These higher order units also have two properties: *object type* (such as *word*), and *string value* (which is the concatenation of the underlying values, such as *ch-ai-r*).

7.2.3 Productions

In the current simulations, only three types of production were required to generate solutions to a range of anagrams: a *cluster former production* (that chunks groups of vowels or consonants into likely sequences), *syllable former productions* (that chunk letters and clusters into syllables and words), and a *termination production* (that determines if the program has generated a solution). Although only these three productions are used, the system exhibits complex problem solving behaviour, utilising FAE's parallel search and backtracking capabilities to explore different letter combinations in generating a valid solution.

The Cluster Former Production

The cluster former production inspects visible objects in the workspace (i.e. objects that have not already been chunked), and tries to combine them into likely vowel or consonant clusters. For example, given the letters *s* and *t*, two competing production instantiations are likely to be triggered; one trying to form the sequence *s*-*t*, and the other trying to form *t*-*s*. Allowable letter sequences are determined from the information stored in FAE's semantic memory. The cluster former production is also capable of chunking existing chunks (such as *t*-*h*), into higher order sequences (such as *(t*-*h)*-*r*), again based upon the information stored in long term memory.

In determining what letter combinations should be investigated, productions react to active nodes in FAE's semantic network. In the semantic network (shown in figure 7.1 above), all the nodes corresponding to structures presently found in working memory receive activation (such as the letters *s* and *t*). This activation spreads to higher order combinations (*s*-*t* and *t*-*s*), to a degree based on their connection strengths (a function of their relative frequencies). The cluster former production retrieves a set of all such activated nodes, and creates a production instantiation bound to each instance of the underlying pairs (i.e. *s* and *t*) found in working memory (see figure 7.2). Only creating instantiations for active nodes reduces the number of computations required, focusing the computational resources of the system to search for likely combinations (rather than looking for the existence of pairs that don't exist). The strength of firing of each instantiation is a function of the memory retrieval strength (the activation of the retrieved node), favouring the construction of frequent combinations of letters.

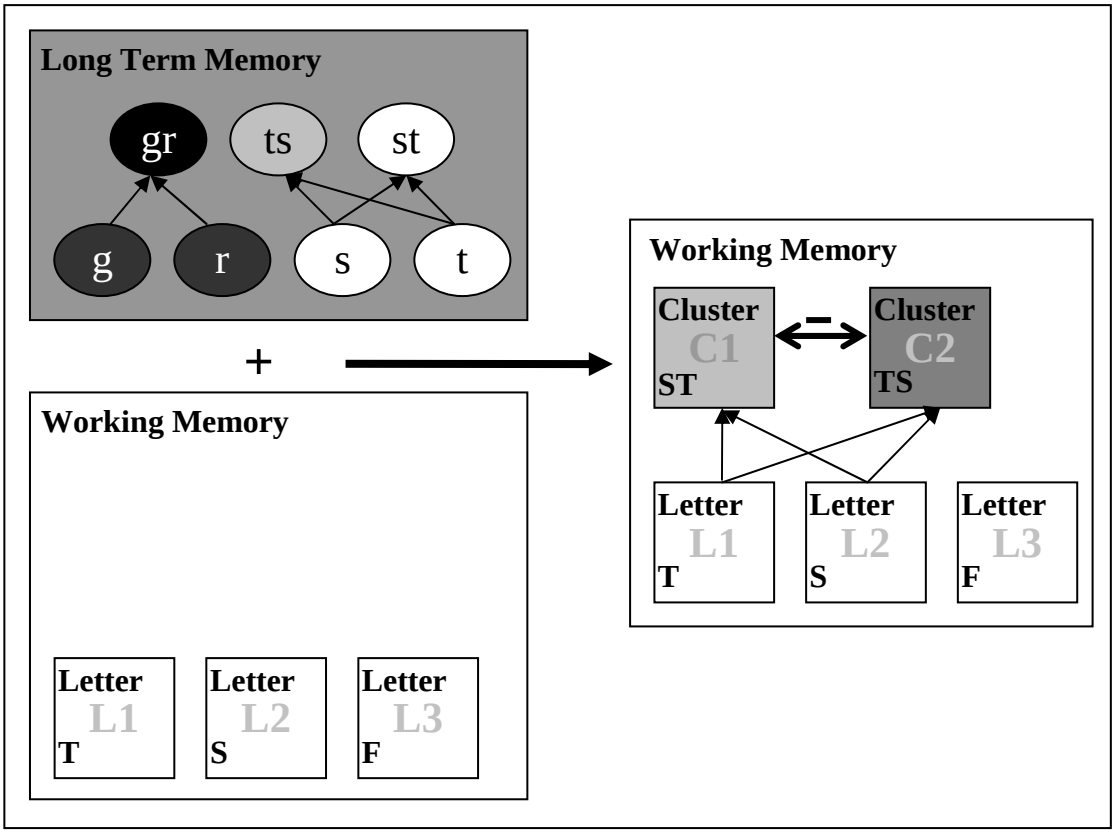


Figure 7.2 The formation of letter clusters in FAE. In long term memory, nodes corresponding to letters found in the workspace will be active, spreading activation to higher-order groupings. Based on these active nodes, the cluster former production will attempt to form these groupings in working memory. Groupings that share underlying objects inhibit each other, resulting to the emergence of mutually exclusive groupings over time.

Syllable Former Productions

In FAE's long term memory, information is also stored about the relative likelihood that each letter, consonant cluster or vowel cluster can be used as the beginning, middle or end of a syllable. Syllables in FAE can take three forms: a complete syllable (consisting of an initial consonant cluster, vowel group, and final consonant cluster), a syllable lacking an initial consonant cluster (e.g. "ing") or a syllable lacking a final consonant cluster (e.g., "the"). Syllables are formed in working memory through the actions of syllable former productions. There are four such productions utilised by FAE, one for each kind of syllable, and an additional production that can group pairs of syllables into higher-order units.

In creating fully formed syllables, the *complete syllable former* production makes three explicit retrievals from memory that return the set of active letter nodes that are linked to the "start", "middle" and "end" properties in long term memory. As a result, a single production instantiation is formed for each unique syllable and is bound to the corresponding objects in working memory (see figure 7.3). The same kind of binding process is used to determine the other forms of syllable. As with clustering productions, any of the proposed syllables that share underlying elements are determined to be incompatible, and are linked through inhibitory connections.

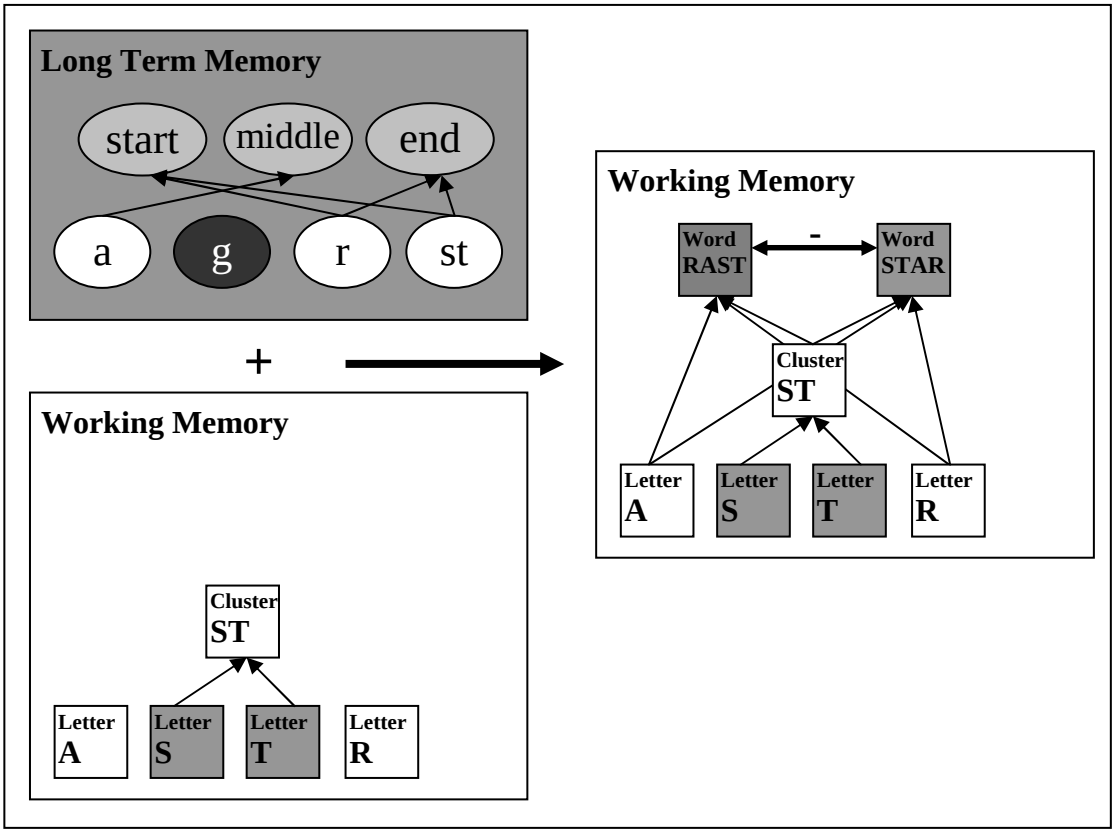


Figure 7.3 The Complete Syllable Forming Production. This production conducts three sets of retrievals from memory, returning the set of active letter strings that can either be located at the beginning, middle or end of a syllable. For each valid combination, a new object is proposed in working memory. Incompatible proposed syllables (i.e. syllables sharing the same underlying elements) are linked through inhibition.

Termination Production

The termination production is used to terminate processing as soon as a valid solution has been found. This production checks for the existence of a "syllable" object in

working memory, and halts the program if there exists no letter, cluster or syllable object that has not been chunked into a higher order structure (i.e. if the selected syllable object is the only ungrouped structure remaining).

7.3 Example Runs

The following sections detail simulations that exemplify the solution formation process in FAE.

7.3.1 Puzzle #1: u d o h l s

The following set of screen dumps demonstrate solution formation of a “simple” anagram in FAE (i.e. a problem in which a solution forms automatically without the need for backtracking).

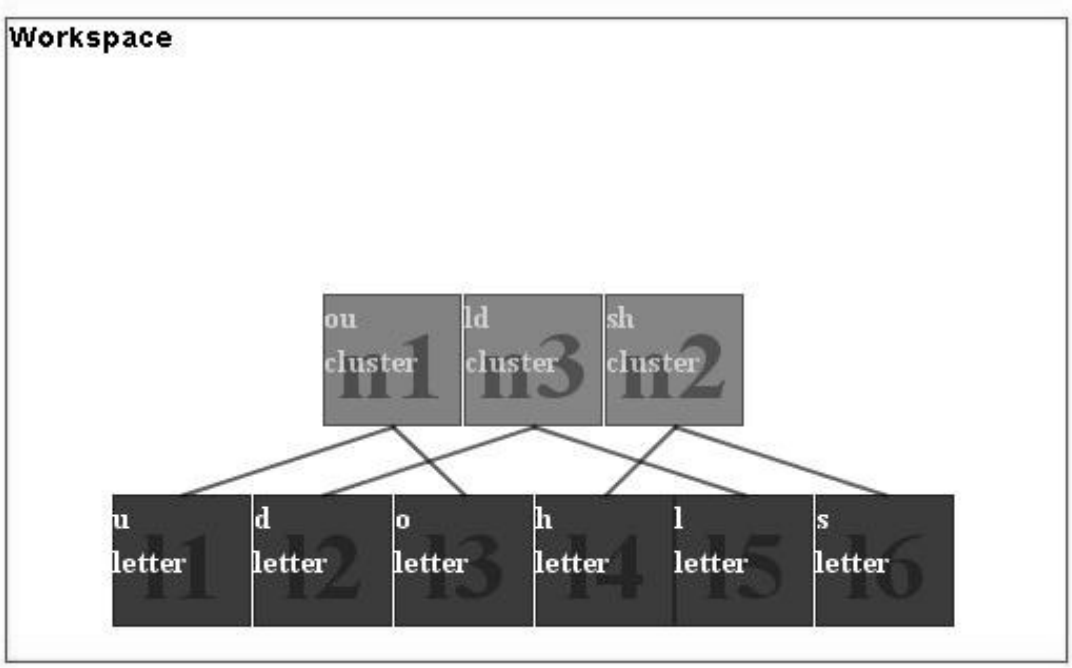


Figure 7.4 Given the initial letters u, d, o, h, l, and s, three letter clusters are proposed automatically by the cluster former production.

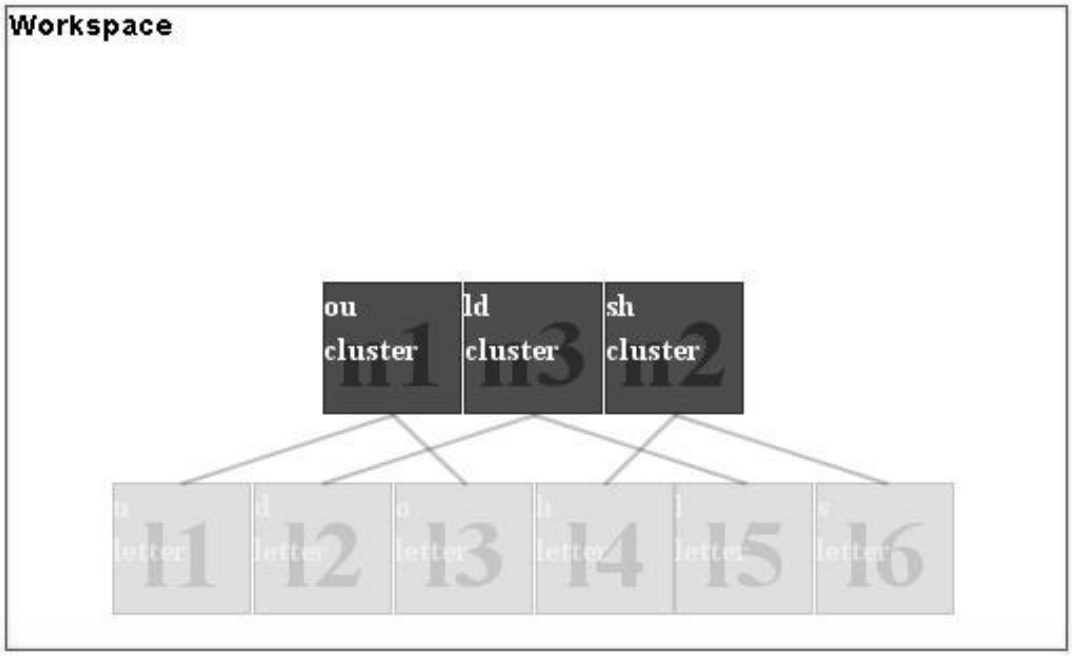


Figure 7.5 As the proposed clusters are orthogonal, all three clusters gain activation over time. Once the clusters reach a threshold activation level, the underlying letters become “chunked”, and invisible to the production system.

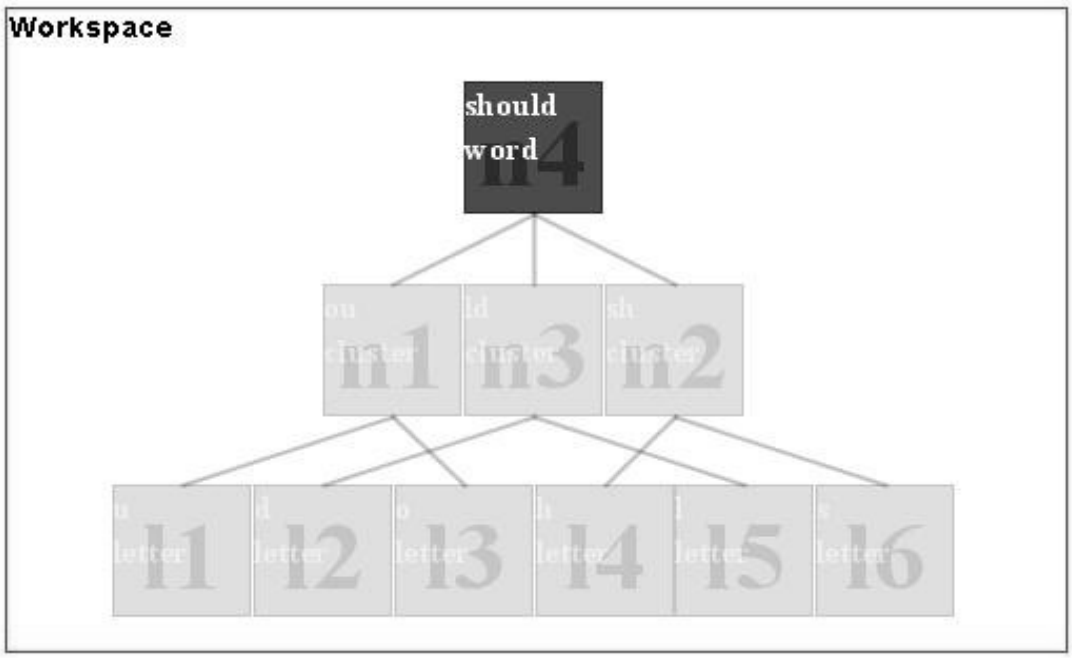


Figure 7.6 According to FAE’s long term memory, the chunk “sh” can either appear at the beginning or end of a syllable, whereas the chunk “ld” can only appear at the end. Thus, joined by the vowel cluster “ou”, the only valid combination is sh-ou-ld, which is constructed by the word-former production, leading to a solution.

7.3.2 Puzzle #2: k t h i n g

The following puzzle represents a more typical example of the creative process in FAE, with the most probable letter combinations leading the system to an impasse. To overcome such problems, backtracking and recombination is required to find a valid solution.

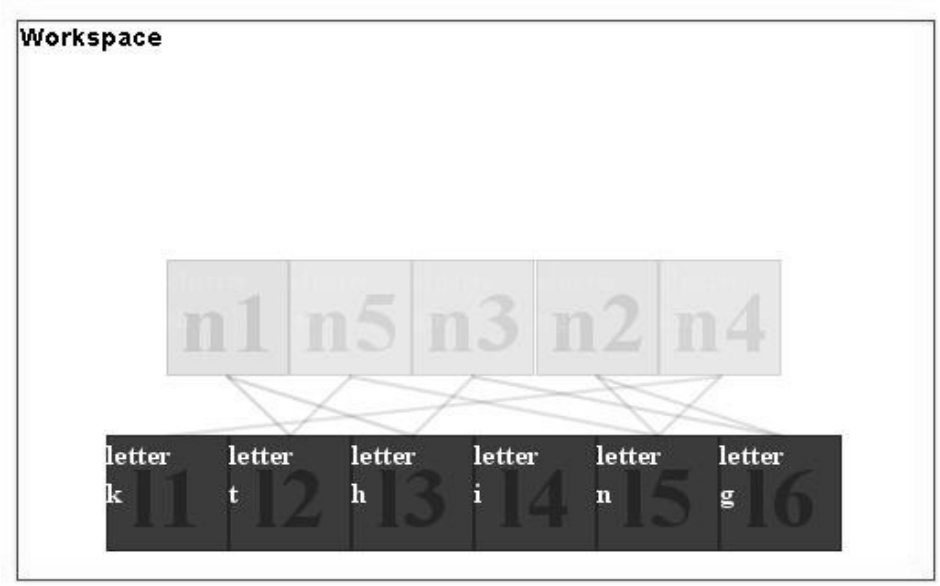


Figure 7.7 Given the initial letters k, t, h, i, n and g, several different consonant clusters can possibly form. Unlike in the previous problem, many of the clusters are incompatible, reusing the same underlying letters. As a result, the structures are linked through inhibition, competing for activation.

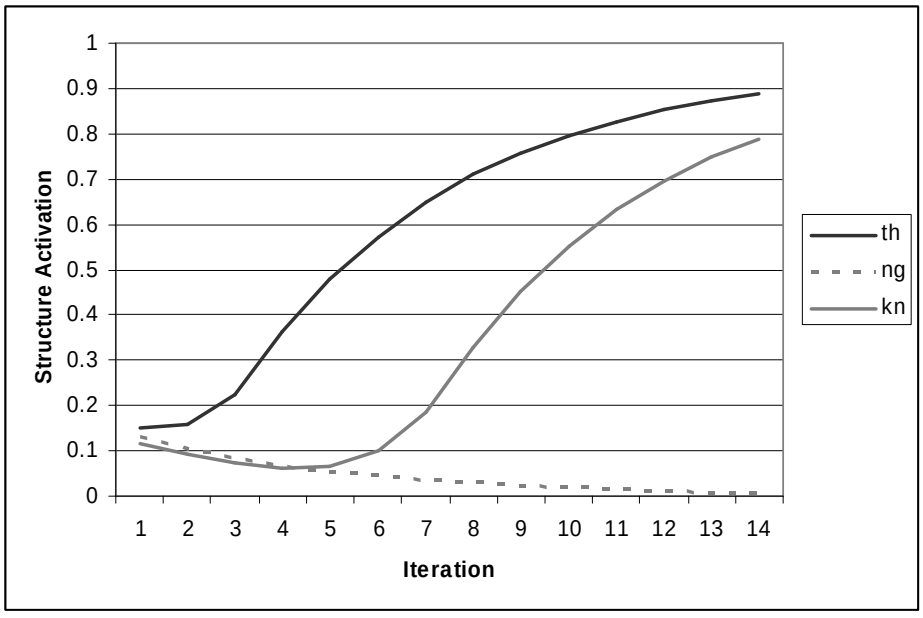


Figure 7.8 With many structures competing for activation, the resulting links act as a parallel constraint satisfaction network, settling into a global solution over time, favouring the formation of more frequent letter combinations.

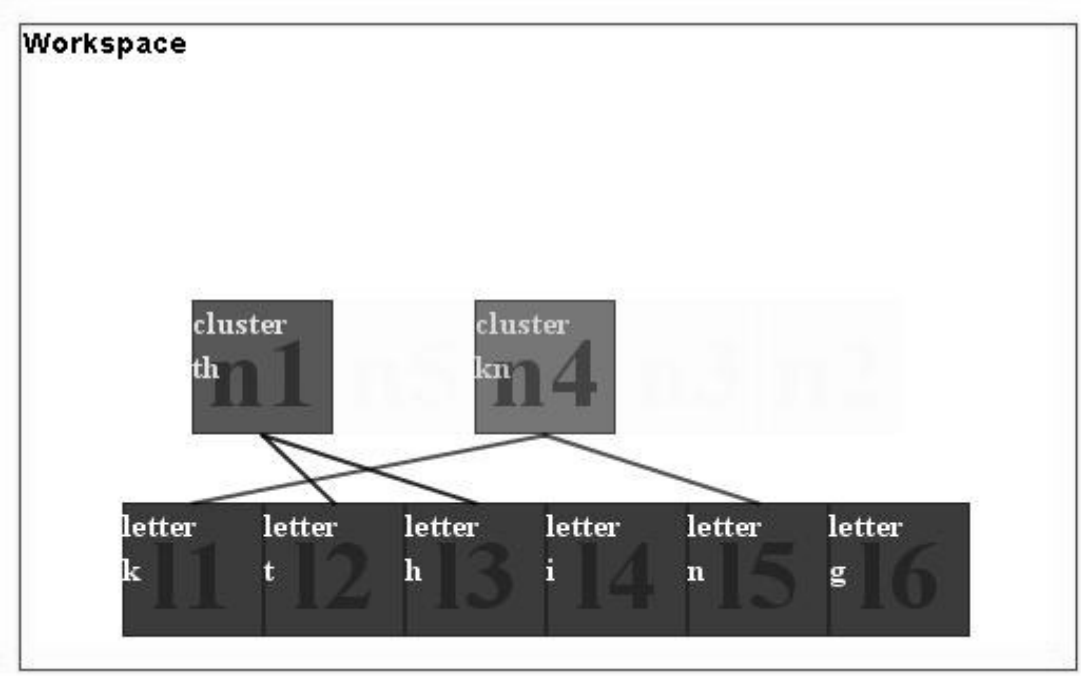


Figure 7.9 Over time, the conflicts are resolved by FAE's underlying attractor network resulting in the formation of the letter clusters "th" and "kn".

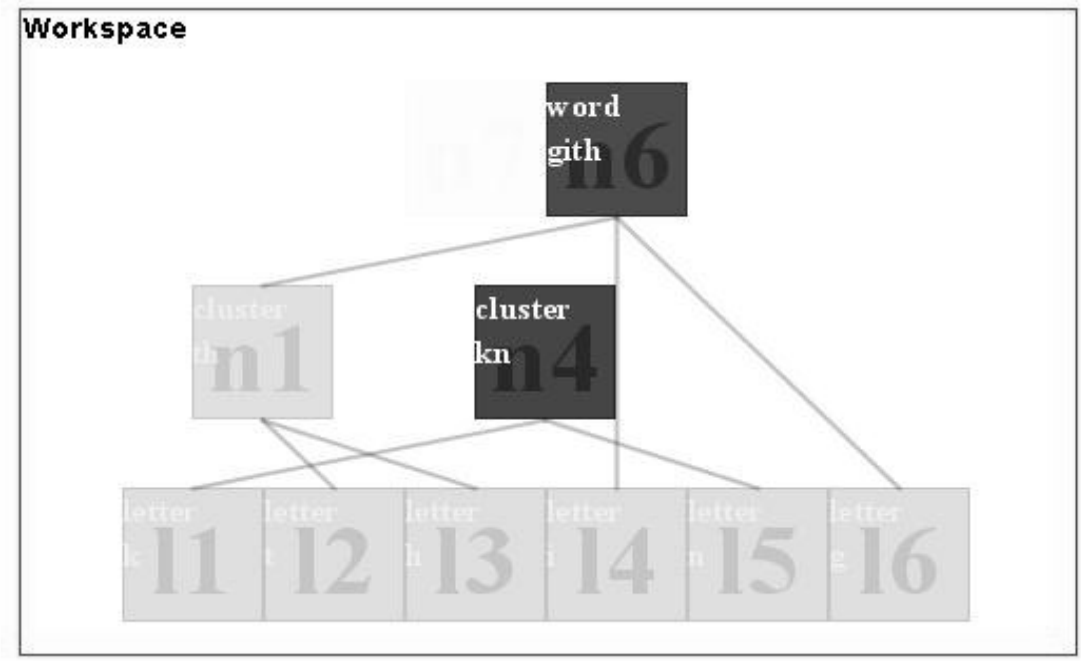


Figure 7.10 Based on considering "g" as an initial start to a word and "th" as a possible ending, the word "g-i-th" is constructed. However, as there remains a letter cluster that has not been used (i.e. kn), the formed combination is not considered a valid solution.

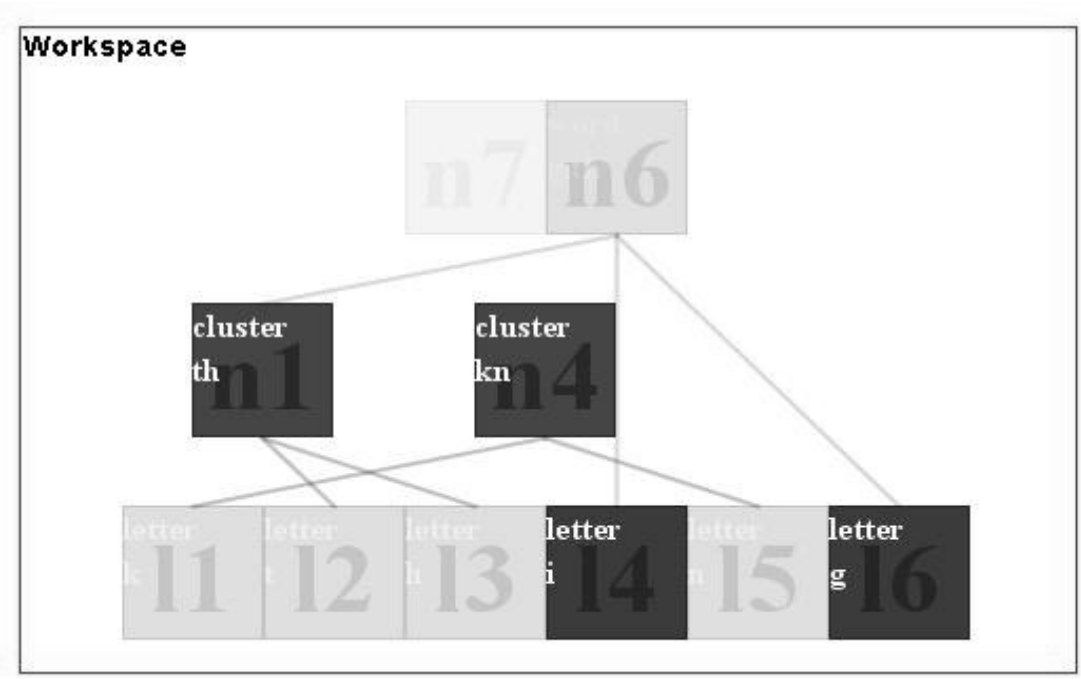


Figure 7.11 As no new structures can form from the previous state, the episodic trace of the situation increases over time. Once a threshold trace strength is reached, inhibition is applied to the Workspace, targeting the word object "g-i-th", and backtracking to the previous state.

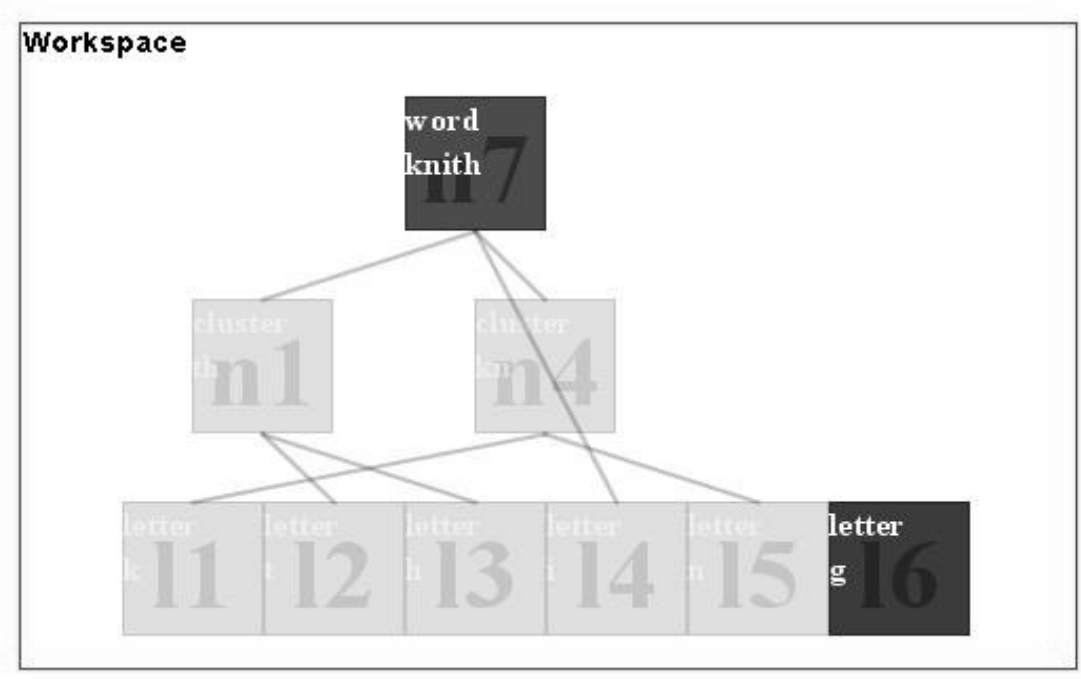


Figure 7.12 Due to inhibition of the alternative combination "g-i-th", other alternatives can arise, leading to the formation of the word "kn-i-th".

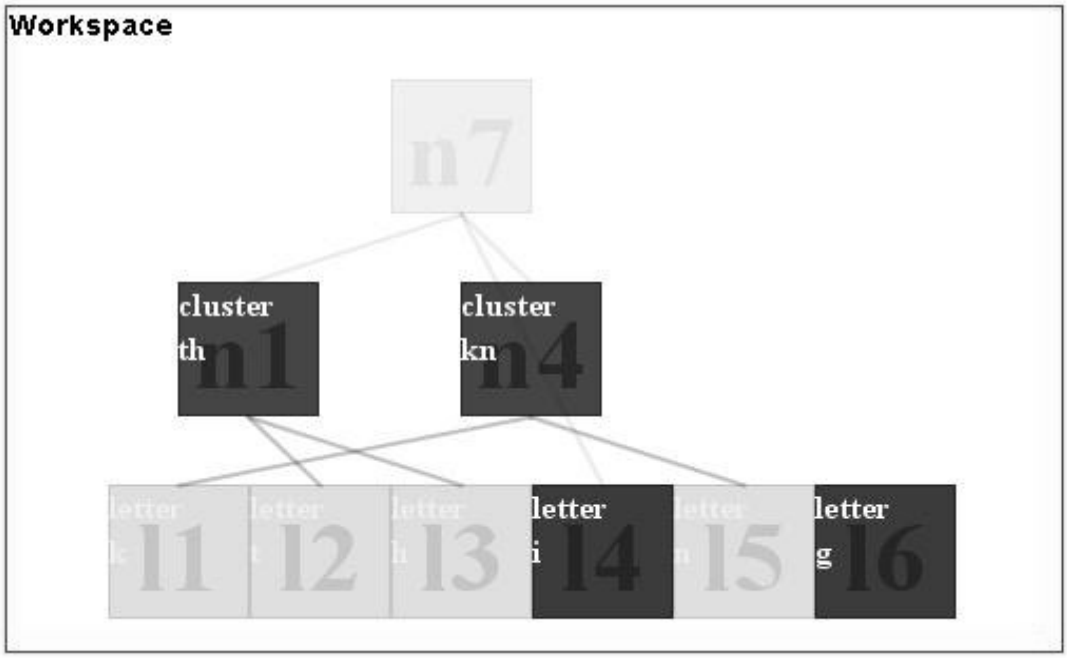


Figure 7.13 Again, as the word “kn-i-th” does not utilise all the letters initially given, it is not considered to be a valid solution. As no new structures can form this state, the episodic trace increases to a threshold level, and the word “kn-i-th” is inhibited.

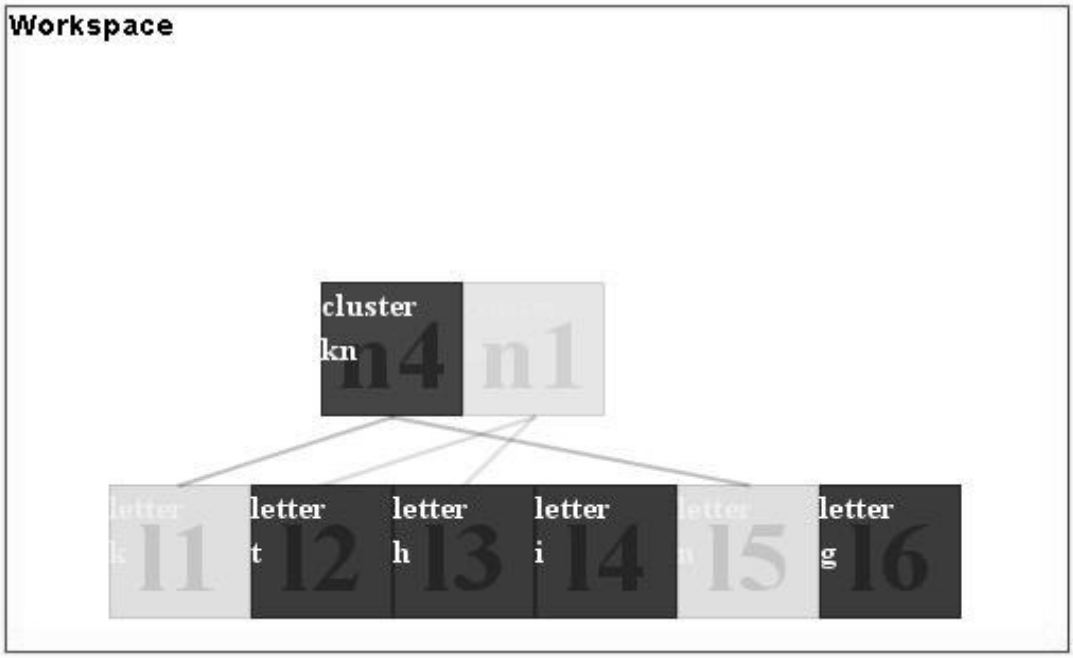


Figure 7.14 Once again, as no new structures can be formed from the current state, further backtracking occurs to allow the exploration of new combinations.

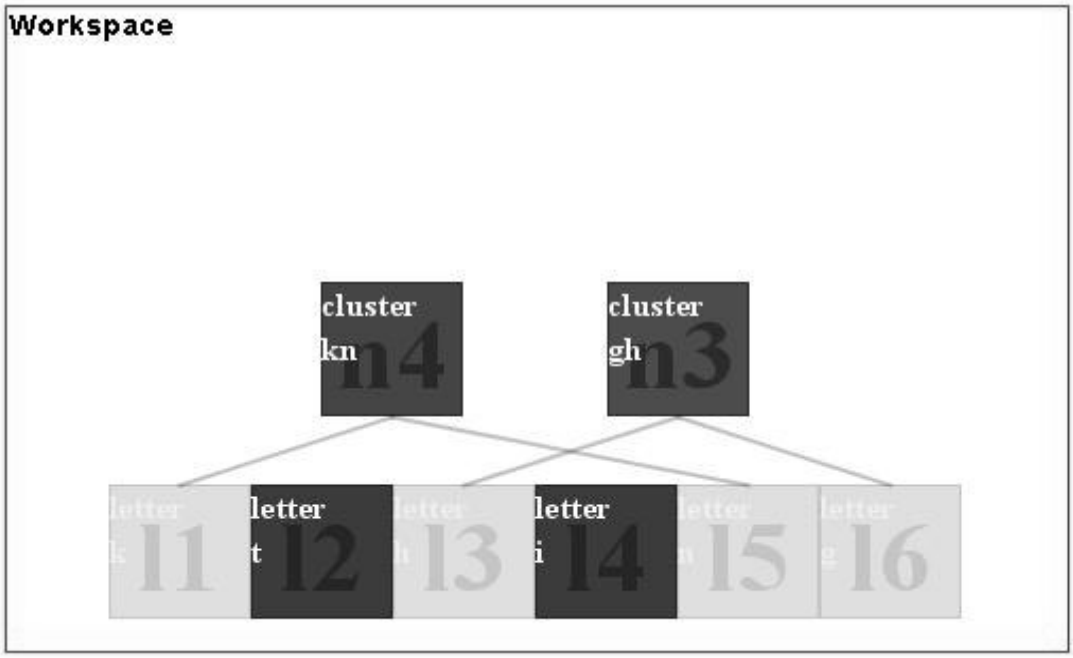


Figure 7.15 As the consonant cluster t-h is destroyed, competing alternatives are no longer inhibited and are permitted to form. In this case, the letters g and h are chunked into the cluster gh.

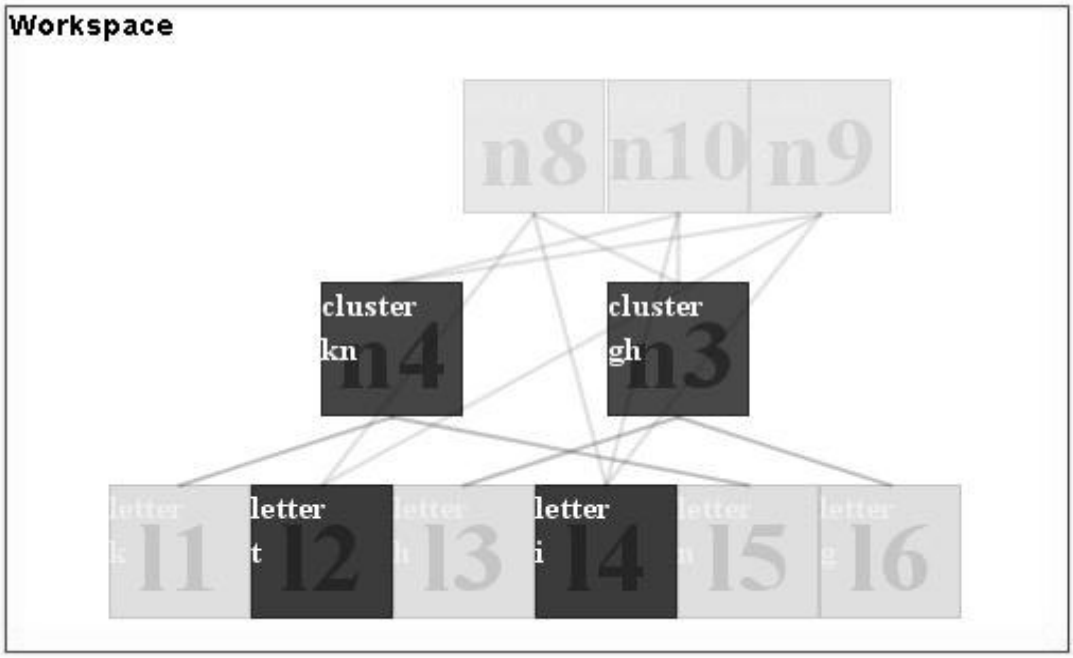


Figure 7.16 From this new state, several overlapping syllables (e.g., kn-i-t) and letter clusters (e.g., gh-t) are possible and compete for resources.

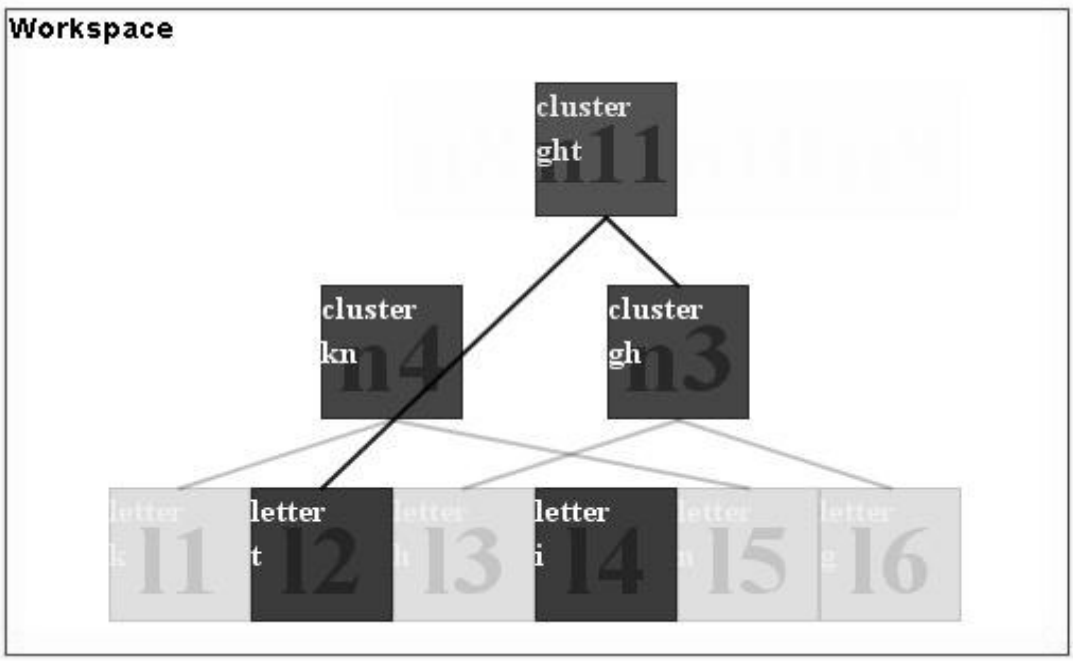


Figure 7.17 Over time, the cluster gh-t gains activation, inhibiting the other possibilities.

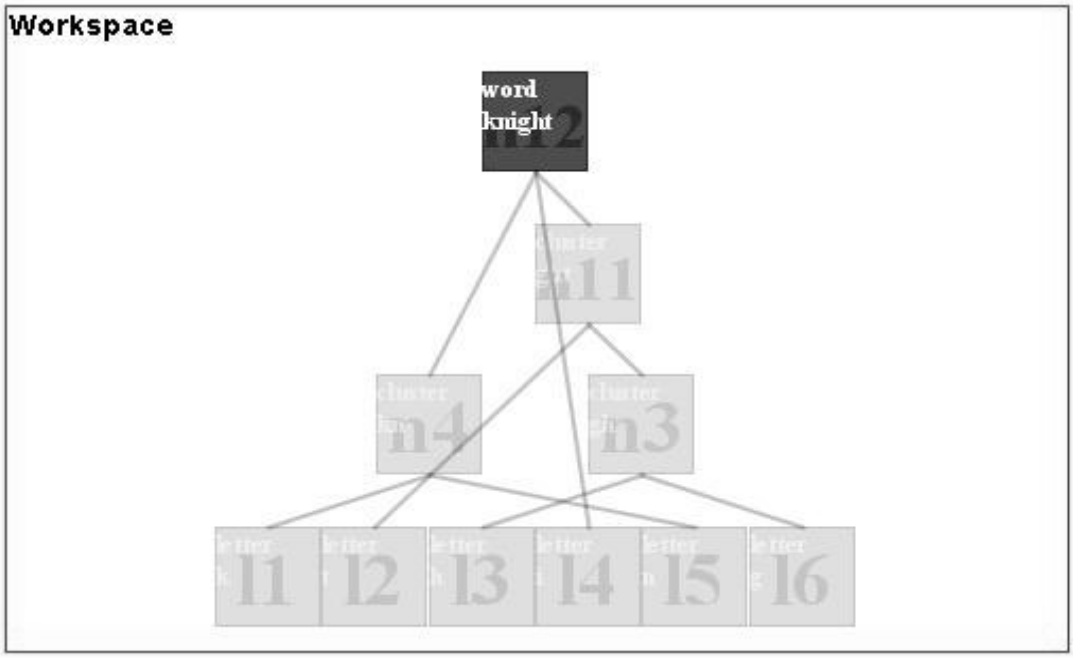


Figure 7.18 Given that the chunks kn and ght are an appropriate start and end to a syllable respectively, the word kn-i-ght is formed. As this word incorporates all the given letters, the system halts with the given solution.

7.4 Discussion

Many instances of creativity can be viewed as the novel combination of familiar ideas (Boden, 1994). In creating word-like candidates in the game of Jumble, such "familiar ideas" include the frequency of letter co-occurrence within vowel and consonant clusters, and the probability that such clusters can occur at the beginning, middle or end of a syllable. This chapter illustrates FAE's capability to utilise such regularities to form word-like candidates. As there is no explicit dictionary, any generated sequence will be "novel" to the system and thus can be viewed of as a "creation".

In generating word-like structures, FAE utilises the regularities mentioned above to group letters into vowel and consonant clusters, clusters into syllables, and syllables into words. In using such a bottom-up approach to word formation, many dead-end paths exist, such as the creation of the word "knith" in problem 2 that leaves the letter "g" ungrouped. In dealing with such impasses, mechanisms for backtracking and recombination are required. In FAE, this is achieved through the use of its episodic memory that inhibits the structures found in impasse situations, and allows the system to backtrack and continue searching. Due to the number of false avenues to explore, the domain of Jumble would prove difficult for other prominent models of cognition such as ACT-R and DUAL which do not have explicit backtracking mechanisms or a memory for visited states.

In lieu of an episodic memory, the original Jumbo program (Hofstadter, 1983) employed two main means of resolving impasses that are dissimilar to the strategies employed by the current approach. Firstly, backtracking could be achieved through the deployment of entropy increasing codelets that would break up existing structures. Although seemingly effective in the domain of Jumble, this strategy is not effective in domains in which action selection is heavily biased, as once a structure is destroyed, it is likely to be rebuilt. As a result, for a weaker action to be considered, the stronger alternatives are likely to have been explored a multitude of times. In the FAE implementation, such re-exploration does not occur, due to the inclusion of an episodic memory that helps lead the system out of and away from impasses states.

In the original Jumbo program, in addition to entropy increasing codelets, "entropy preserving" codelets were also used to explore the search space. Such codelets did not destroy structures, but rather, were capable of rearranging them. For example, codelets could be used to regroup the word "pang-loss" into the interpretation "pan-gloss", or to reverse the syllable "stan" into the syllable "nast." Such rearrangements could often lead the system uphill, to a more satisfactory solution. In the current implementation in FAE however, there is no such equivalent process. In this approach, when letters are chunked into higher order structures, the underlying units are invisible to productions, making it impossible to restructure them. However, if the program was not asked to halt at the formation of a solution, many of these regroupings would be explored automatically. For example, if the system did not halt at the formation of "stan", but rather, noted that it was one of several possibilities, this structure would then be inhibited, leaving the underlying elements to reform into "nast." This approach is analogous to serially evaluating the potential next moves in a game of chess. A beneficial extension to the architecture of FAE therefore, would be the ability to compare such explored solutions, and to select the best option. This is not necessarily a large extension to the program, as previous states are already stored in episodic memory. In comparing solutions, an intrinsic evaluation also exists in the form of the activation of the top-level chunk. That is, in the Numble implementation, as productions fire with a strength related to the frequency of the letter combination, the "better" solutions will have higher resulting activation value which could be explicitly used to compare each answer.

Despite successfully producing answers to a range of anagram problems, several limitations of the approach taken by FAE should be noted. In the original implementation of Jumbo, syllables could either be complete (consisting of an initial consonant cluster, a vowel group, and a final consonant cluster), or could be missing either an initial or final consonant cluster. Although this was also implemented in FAE, the productions creating the latter two structures were disabled in the example problems described above. This was not because FAE failed to derive a solution with their inclusion (in fact, the same final answers resulted), but rather, because their inclusion generated many more false solutions for which FAE would spend time exploring. For example, in problem #2 (the anagram *k-t-h-i-n-g*), the inclusion of these incomplete syllable forming productions would allow the syllables "ki", "ti",

“hi”, “ni”, “gi”, “it”,”in”,”ig” to be initially considered. For each one of these options, a new object would be created in the workspace, linked to the other alternatives through inhibition. This would create quite a large number of proposed structures, when taken with all the possible syllable clusters (e.g., “nit” and “hig”) and consonant clusters (e.g., “ng” and “th”). Rather than being an inherent limitation of FAE, this seems to be more related to the way in which the problem of Jumble was approached. Allowing all possible syllables and consonant clusters to be considered in parallel evidently leads to a broad initial search. A better approach may be to pose the problem as a more serial exploration, focussing on an initial starting letter, and growing the string a letter at a time, only investigating what letters are likely to be next in the sequence. From introspection, this alternative seems plausible. To reduced the combinations explored in parallel, heuristics could also be used to narrow down the search space. For example, it seems unreasonable to form a syllable (such as “knith”) if there are consonants left ungrouped. Such heuristics could be encoded within productions to limit the number of possibilities explored both in series and in parallel.

A second limitation of the simulation described in this chapter is that it “created” words in a purely bottom-up fashion. It is unlikely however, that this approach would generalise well to many other examples of creativity. For example, it is unlikely that a coherent piece of music could be generated from the ground up, starting with notes that frequently co-occur. Instead, in such instances, there are clearly higher order structures (such as bars, chord progressions, stanzas etc.), that create explicit contexts that affect the acceptability of each note. It is evident that the creative process requires the integration of both top-down and bottom-up processes, utilising such strategies as multi-level cleaving, splicing, regrouping, reordering and rearranging to improve the quality of the work (i.e. the adherence to multi-level statistical regularities). Although FAE is capable of integrating bottom-up and top-down pressures (as demonstrated in the previous two chapters), evaluating how these mechanisms can be utilised within the creative process is a matter for future work.

7.5 Summary

This chapter described an application of FAE to the problem of anagram formation. This process can be viewed as creative, as humans typically generate word-like candidates that they then check against an internal lexicon for validity. In the creation of such candidates, statistical regularities (such as frequencies of letter combinations within vowel and consonant clusters) are exploited to help generate only likely alternatives. As was demonstrated in this chapter, FAE is able to perform the creative task of anagram formation adequately, using the same set of mechanisms that it used for planning and flexible perception. The strategy employed however, is mostly bottom-up, and could be improved through the integration top-down factors that play an important role in other examples of creativity such as in the composition of music.

Chapter 8
Modelling Analogy-Making

Analogy-making, or the ability to perceive non-identical objects or situation as being the same, is a process central to human cognition that plays a fundamental role in recognition, categorisation, learning, problem solving and creative thought (Holyoak, Gentner, & Kokinov, 2001; Hofstadter & FARG, 1995). For example, even a fairly low-level tasks such as the ability to recognize a face from different perspectives can be viewed as a form of analogy-making as it requires the equating of non-identical features at some high level of abstraction. Apart from recognition, drawing an analogy can also be useful in generalising information across different situations. For example, when faced with a stray dog, retrieval of a scenario in which a canine with similar properties was aggressive may infer that the current instance may also have the same tendencies and should be avoided. Due to the ubiquity of analogy-making and analogical reasoning in human thought, it is an important process to capture in general models of cognition. This chapter explores the ability of FAE to account for such processes.

8.1 Cognitive Processes Underlying Analogy-Making

According to Kokinov and French (2003), analogy-making incorporates the following important sub-processes:

(1) Representation-building. For a situation to be mapped to a similar scenario, it needs to be "perceived." That is, a high-level representation describing the salient objects and relationships needs to be formed in working memory.

(2) Retrieval. Although analogies can be drawn across two situations in the immediate environment (such as formulating a solution to the problem A:B::C:? found on IQ tests), in many instances, the target analog needs to be explicitly retrieved from memory. According to Kokinov and French (2003), such retrieval is mostly based upon superficial similarities in the objects, properties and themes found across the two scenarios.

(3) Mapping. The most fundamental aspect of drawing an analogy is determining what aspects of two scenarios correspond to each other. Generally speaking, such mappings are based on similarities in the role that each entity is playing within the situation. For example, in drawing an analogy between the Solar System and the Rutherford atom, the sun is mapped to the nucleus, not because of physical similarities, but because they both have other entities rotating around them.

(4) Transfer. An important use of analogy-making is in generalising information from other situations. "Transfer" is the process of drawing on information from one domain or situation and applying it to another (such as inferring that a novel instance of a dog may be dangerous).

(5) Evaluation. This is the process of determining whether or not the transferred knowledge is appropriate for the new domain.

(6) Learning. In contrast to the retrieval of a specific episode, much knowledge can be inferred from the regularities found across a number of situations. Such abstractions can be viewed as a form of "learning."

Based on the sub-processes mentioned above, it can be assumed modelling analogy-making incorporates at least four out of the six cognitive abilities specified in chapter one. These include the ability to select and attend to relevant aspects of the problem (i.e., there are many different features on which mappings can be based), the ability to process information in a contextually sensitive manner (i.e. mappings are context dependent), the ability to create and manipulate complex relational structures (which is required for representation-building), and the ability to exhibit rational goal-directed behaviour (i.e. the goal being to draw an analogy).

8.2 The FAE Implementation of Analogical Reasoning

In the literature, there are no models of analogy-making that explore all of the above listed sub-processes. For example, the Fluid Analogy models of Copycat, Tabletop and Metacat do not explore the issues of learning or the retrieval of similar scenarios from memory (Mitchell, 1993; French, 1995; Marshall 1999). However, these models do differ from other prominent models of analogy-making in that they assume that representation formation and mapping are parallel and co-dependent processes. As argued by Hofstadter & FARG (1995), this co-dependency is an essential property of cognition, as how we perceive a situation (what representations come to mind) is often dependent on the task that is being performed or the analogy that is being drawn.

In exploring analogy-making in Copycat and Metacat, a domain of letter-string analogy problems was used (described in chapter two). This domain is useful as apart from requiring representation formation and mapping, the processes of transfer and evaluation can also be explored. That is, given a problem such as abc:abd::ijk:?, in the generation of a solution, mappings need to be made between entities based on similarities in perceived relationships, followed by a transfer of the initial transformation onto the target domain. In Copycat and Metacat, an evaluation of the proposed solutions could be inferred from the final temperature value (a measure of structural coherency). Due to the utility and simplicity of the letter-string analogy domain, FAE's ability to perform representation formation, mapping and transfer will also be assessed using these problems.

8.2.1 FAE's Long-Term Memory

In modelling analogy-making in the letter-string domain, FAE's long term memory stores the definitions and relationships between concepts that are required for forming the appropriate representations in working memory (see figure 8.1). In particular, objects in working memory can be assigned an *object type* (either *letter* or *group*), a *letter category* (*a-z*), a *position* in the string (*leftmost*, *rightmost* or *middle*) and possibly an associated string *length* (*1-5*). Relationships between objects include

"successor", "predecessor", "sameness" and "different." With the exception of the "different" relation, all the concepts in the FAE implementation are the same ones utilised by Copycat (see chapter two for details). How the FAE implementation also differs from Copycat is that the semantic network does not have slip links between concepts (i.e. denoting allowable conceptual mappings), and that the successor and predecessor links are weighted differently, favouring the perception of successor relations in working memory. This weight bias is meant to reflect the relative exposure to successor relations (i.e. the alphabet is practiced more frequently forwards than backwards).

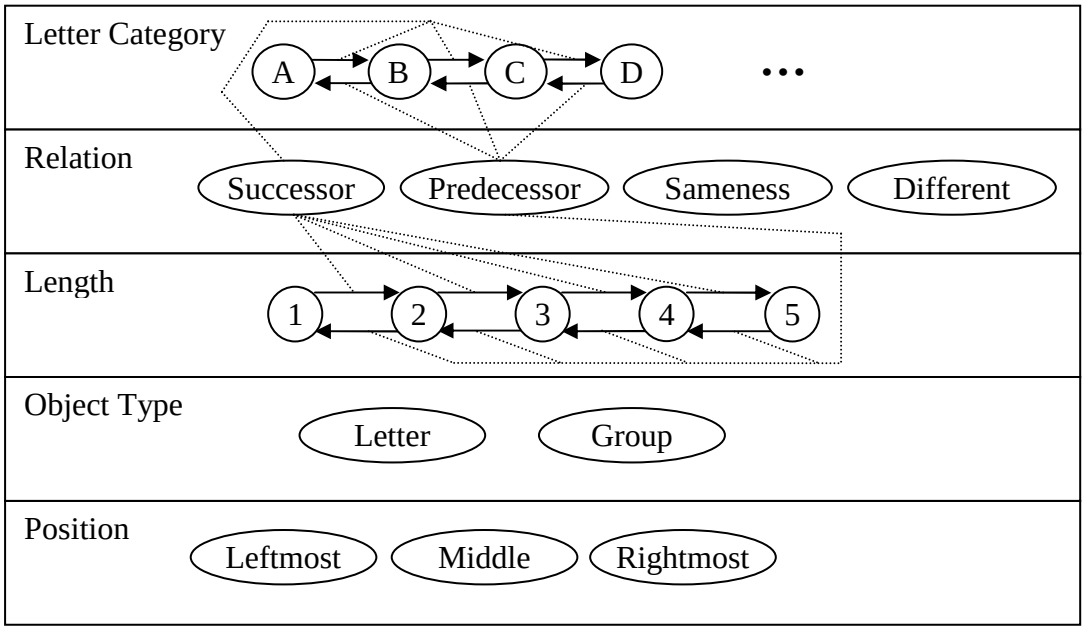


Figure 8.1 Concepts in FAE's long-term memory. The concepts in the FAE's semantic network (with a few omissions) are identical to those utilised by Copycat. The most important way in which this information differs from Copycat is with the omission of explicit "slip links."

Instead of incorporating explicit links in the semantic network that denote potential conceptual slippages (concepts that can be equated across domains), in FAE, such information is stored implicitly through the representations of the concepts themselves. That is, concepts are stored as distributed binary patterns, with those that can be possibly equated having overlapping features. For example, the concepts "leftmost", "rightmost" and "middle" are represented by the vectors, [1,1,0,0,0], [0,1,1,0,0], and [0,0,0,1,1], making it possible that objects at opposite ends of the string could be equated, but unlikely that an object at the end of the string will be

equated with one in the middle based on spatial position alone. It is assumed in this simulation that the process of mapping can also be achieved using the same local context-dependent processes utilised by the word superiority effect simulations from chapter five.

8.2.2 Working Memory

The structures formed in FAE's working memory are almost identical to those utilised by Copycat: letters can be assigned a spatial location within the string (i.e. *leftmost*, *middle*, or *rightmost*); adjacent objects can be linked through *successor*, *predecessor* or *sameness* bonds; objects can be chunked into higher order groups (such as *sameness groups*); and correspondences can be found across the various strings. The FAE implementation does differ from Copycat however, in the way that information is transferred to the target domain. In Copycat, an explicit rule is constructed in the form "replace X of Y by Z" (such as replace the *letter category* of the *rightmost letter* by its *successor*). In order to generate a solution, a modified rule is then applied to the target string that takes into account the conceptual slippages that have been found in working memory (e.g., the rightmost letter in the source analog, may correspond to the leftmost group in the target string). In FAE, no such explicit rules are required or utilised (as explained below).

In creating a solution string in the FAE implementation, missing information is transferred across all corresponding object pairs without the need for an explicit rule. For example, in the problem abc:abd::ijk:?, the initial letter *a* will be linked to the second instance of *a* through a "sameness bond". As the initial *a* will be perceived as corresponding to *i* in the target string, in the formation of a solution, a new object will be created that shares the initial relation (that is, it is assumed that *i* should be linked to an object that is "the same"). Thus, through this transformation, it can be inferred that the letter "i" is an appropriate initial start to the solution string. For the last letter in the sequence, as *c* is likely to be mapped to *d* through a "successor bond", the letter *k*, will also be mapped to its successor, resulting in *l* being proposed as the final letter (i.e. leading to the solution "ijl") (see figure 8.2).

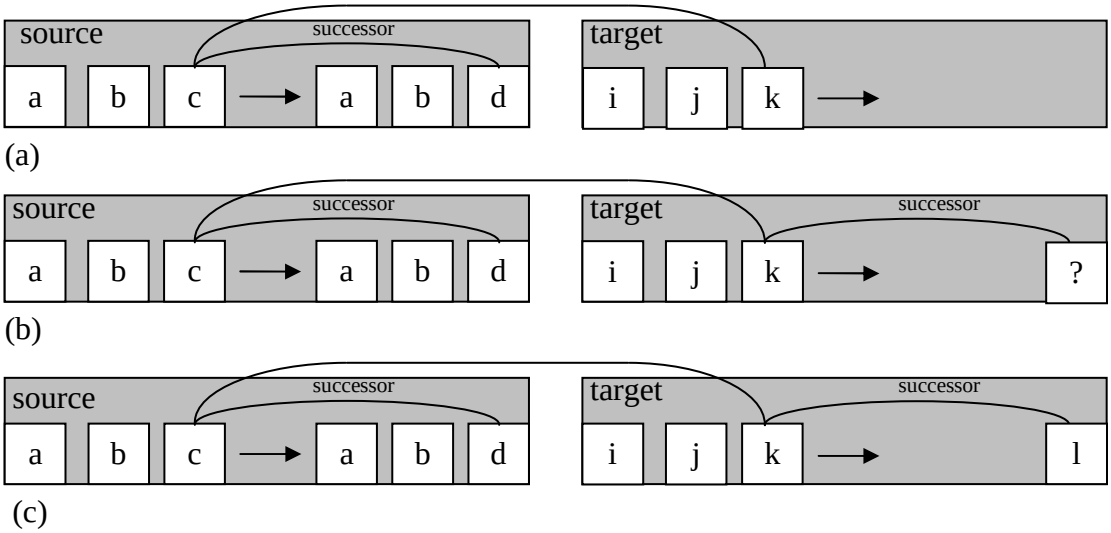


Figure 8.2 The transference of information. (a) Given the problem abc:abd::ijk:?, the letter "c" will be linked to "d" through a "successor bond" and to "k" through a correspondence. (b) In transferring the missing information to the target domain, a new object will be created that is linked to "k" though a similar bond. (c) The descriptions of the object will then be calculated based on the bond (i.e. creating l as the new letter category as it is the successor of k). The same transformations will occur on the other objects, leading to the transferred solution "ijl."

As with Copycat, the solution "ijd" can also be generated for the problem abc:abd::ijk:?. In this case however, *c* and *d* need to be linked through a "different" rather than "sameness" relation (in the simulations, these two descriptions compete with each other). As there is no unique object that is "different" to *k*, the original information will be mapped across directly, leading to the solution *ijd*.

Within the transfer process, conceptual slippages can also occur if they are justified. In FAE, a conceptual slippage is noted if there are differences in the relations between mapped objects. For example, in the problem abc:abd::kji:?, if the initial string is perceived as a group of successive letters to the right and the target string is perceived as a group of predecessors to the right, the initial letters *a*, *b*, and *c* are likely to be mapped to the letters *k*, *j*, and *i* respectively. In this case, the relationship between *a* and *b* is deemed to be equivalent to the relationship between *k* and *j* (as the underlying pair of objects is mapped), leading to the slippage from the concept "successor" to "predecessor". Thus, in the same way that *c* is replaced by its successor in the source analog, *i* will be replaced by its predecessor, leading to the solution *kjh*.

8.2.3 Productions

In the current simulation, there are five main types of productions used: property formers (that perceive such features as spatial location and length), bond formers (that perceive the relationships between adjacent objects), group formers (that chunk objects into higher-order groups), correspondence formers (that form mappings across the domains), and transfer productions (that transfer information between the source and target domains). These productions are described in more detail below.

Property Former Productions

In the letter-string analogy domain, letters are likely to be mapped based on contextual similarities rather than on alphabetic equality. In Copycat, one of the basic contextual cues used for mapping is spatial location within a string, with objects capable of being assigned explicit spatial descriptors if they were found to be at the leftmost, middle or rightmost positions. In accordance with this representational scheme, in the FAE implementation, there are three productions that explicitly tag objects perceived to be at these locations. Using these productions, objects gain a "leftmost" or "rightmost" location property value if they are at the ends of strings, whereas an object gains a "middle" descriptor only if it is adjacent to both a "leftmost" and "rightmost" object.

Bond Former Productions

In accordance again with the representations used by Copycat, Bond Former Productions form links between adjacent objects perceived to be successors, predecessors or identical in letter category. In the FAE implementation, there are three forms of production that are used for this purpose: right-going bond formers (that detect either predecessor or successor relations), left-going bond formers (that again look for predecessor and successor relations), and bidirectional bond formers (i.e. looking for "sameness" relations). These productions have different relative strengths, with bidirectional bonds likely to be perceived relatively quickly (as "identity" is a salient relation), and right-going bonds being preferred over left-going

bonds (reflecting the bias of English speakers to read from left to right). Right-going and left-going bonds are deemed to be incompatible, with the relationship between letters in the sequence *a-b*, either being perceived as a successor to the right, or a predecessor to the left.

In judging the existence of a relation, left-going and right-going productions note the letter categories of adjacent objects, and make an explicit retrieval from memory for a "relation" using these properties as cues. Such retrievals will return either "successor" or "predecessor" (or will fail if no relation exists), with the corresponding bond being proposed. As in long-term memory, successor links are stronger than predecessors, successor bonds are more likely to form in working memory.

Due to the preference for right-going bonds over left-going bonds, and successors over predecessors, the string *a-b-c* is likely to be unambiguously perceived as a group of right-going successor bonds. The string *k-j-i*, however will be ambiguous, either being perceived as containing right-going predecessors, or left-going successors. Such differences in bond direction will result in different objects being mapped across analogs, and different solutions being generated (e.g. *kjh* vs *lji*) reflecting human performance (Mitchell, 1993).

Group Former Productions

Group former productions are capable of taking a sequence of connected objects that share a common bond (such as sameness) and chunking them into a higher order unit with its own set of properties. For example, in the string "iijjkk", identical letters will be linked with the "sameness" relation and are likely to be chunked into "sameness groups." Such objects can take on their own set of properties (such as location and letter category), and be bonded with other structures.

Correspondence Former Productions

Correspondence former productions create mappings between objects across the domains. Such mappings are based on the similarities found in the properties

assigned to each object, such as letter category and string position, but are biased towards the higher-order contextual properties (i.e. string position). As the concepts "leftmost" and "rightmost" are similar (i.e. have similar distributed representations), such mappings are possible, but not probable under most conditions.

As with Copycat, the processes of mapping and representation-building occur in parallel and can be viewed as co-dependent. In the FAE implementation, there are various forms of competition and support that lead to the emergence of a coherent solution. Firstly, each object in the source domain can only correspond to a single object in the target domain (although such an object may represent several underlying letters, such as the sameness group "i-i"). This restriction leads to competition between the various mappings that could stem from a single object. Secondly, adjacent objects can either be linked by a left-going bond, or a right-going bond, but not both. And thirdly, the strength of a bond gains additional support if its two objects are mapped to other objects that are bonded together in a similar way. For example, in the problem abc:abd::ijk:?, if the letters in the same spatial location are mapped, and the sequence a-b is linked by a successor bond, the sequence i-j will have support for forming a successor bond, but no support for forming a predecessor bond in the opposite direction. As described in chapter four, incompatible structures are linked through inhibition, and compatible structures can facilitate each other, creating an overall parallel constraint satisfaction network that will settle into a globally consistent interpretation over time.

Information Transfer Productions

Once the mapping and representation-formation processes have finished (i.e. there is no change in the activation levels of the structures involved), an information transfer production creates new objects in the target domains that correspond to the solution string. As described previously, the properties of these new objects (e.g., their letter categories) share the same types of relationships found in the initial source transformation (taking into account conceptual slippages).

8.3 Example Runs

The following sections describe three of the five main problems explored by the Copycat project, displaying how the same types of solutions can arise in the FAE implementation.

8.3.1 Problem #1. abc:abd::ijk:?

This problem represents a simple example of representation formation and transfer of knowledge across domains. The formation of a solution in FAE is described in figures 8.3-8.7 below.

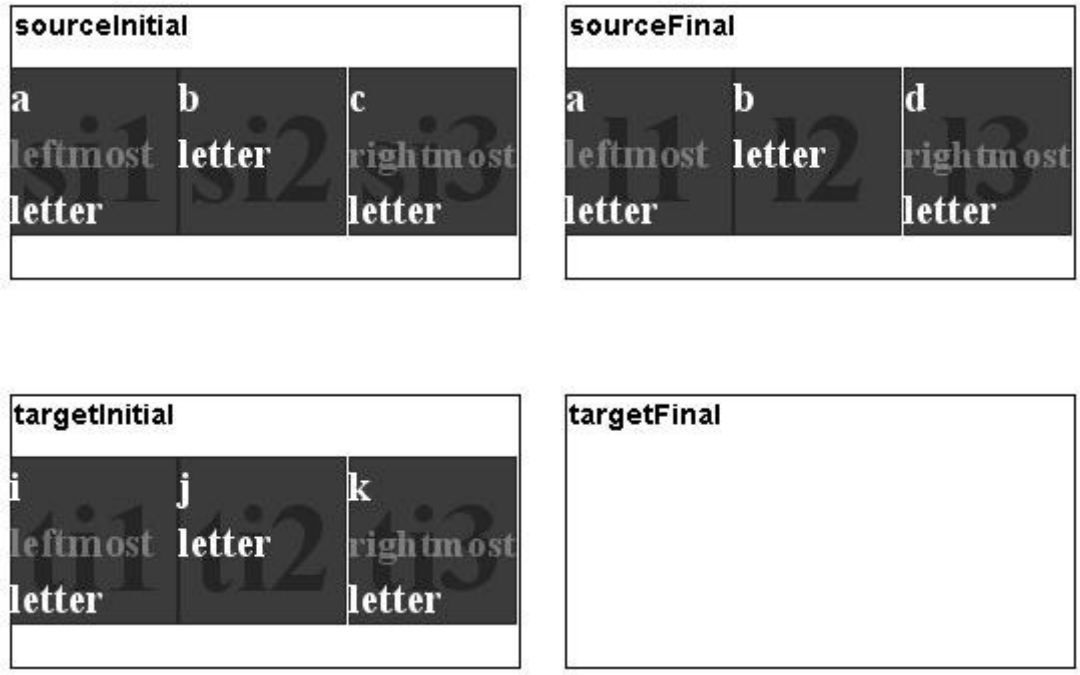


Figure 8.3 In the FAE implementation, the initial task is to assign spatial location properties to the objects, as it is upon such features that mappings are mostly based. In the above problem, the letters at the end of strings are assigned a “leftmost” or “rightmost” descriptor.

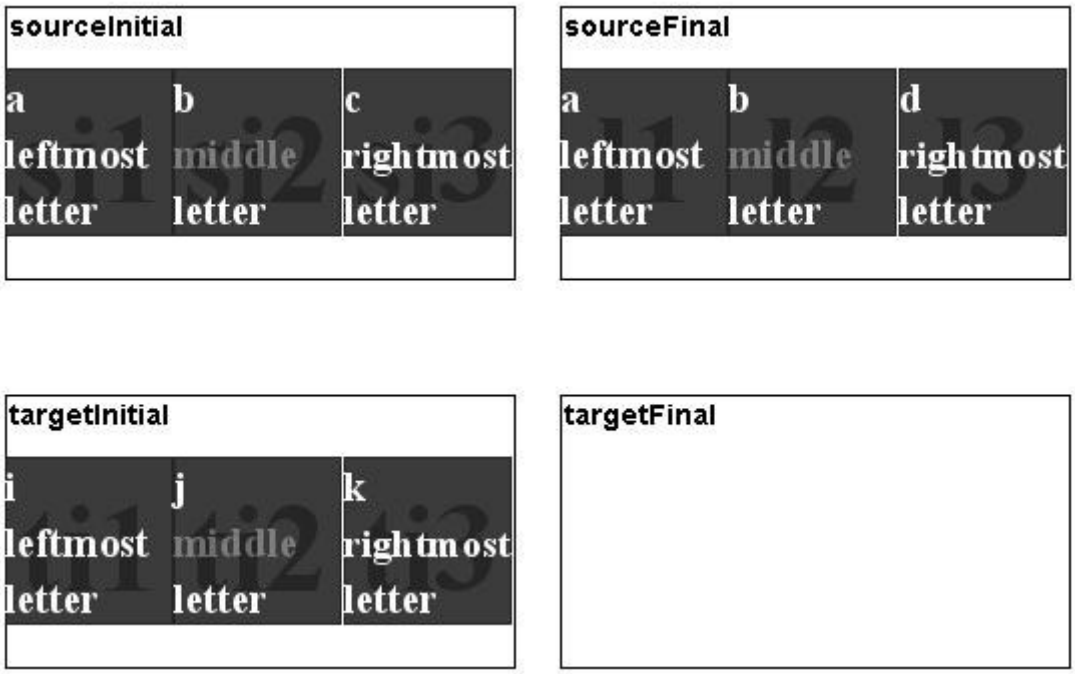


Figure 8.4 Once the letters at the outside location of strings have gained the descriptors "leftmost" and "rightmost", letters that lie directly between these objects can be perceived to be in the "middle".

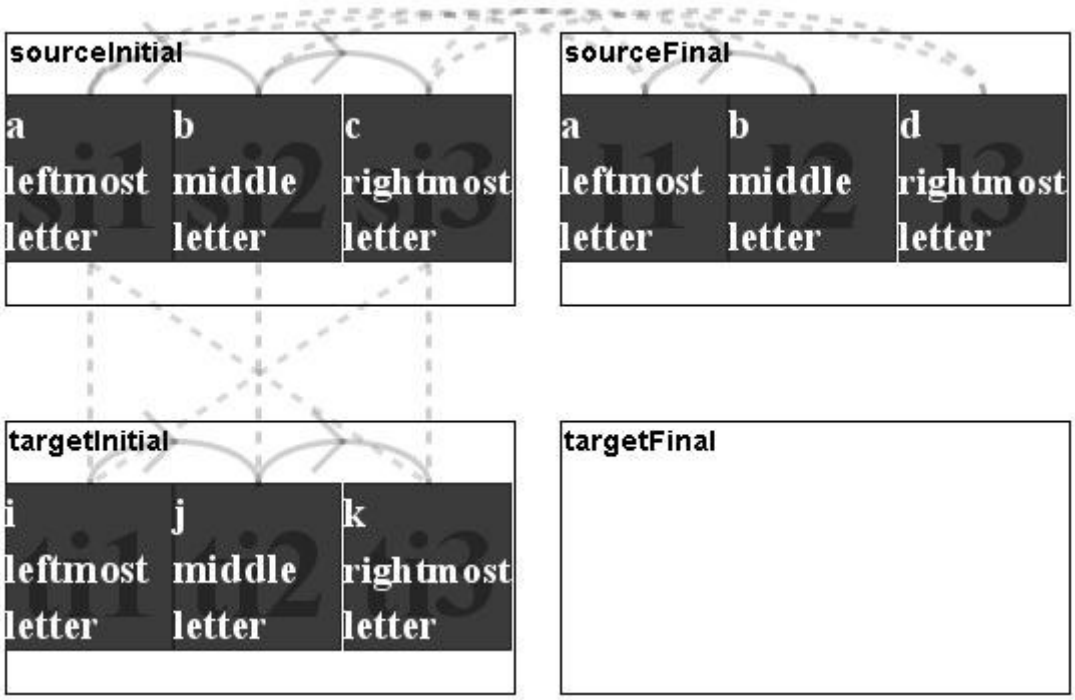


Figure 8.5 Based on similarities in spatial location, correspondences can form between objects across strings. As the concepts of leftmost and rightmost are similar, objects in opposite spatial locations can be mapped, although there is a higher preference to map objects in the same spatial location. At the same time as correspondences are perceived, relationships between adjacent letters are also noted. Both these processes occur in parallel, supporting compatible interpretations, and inhibiting incompatibilities.

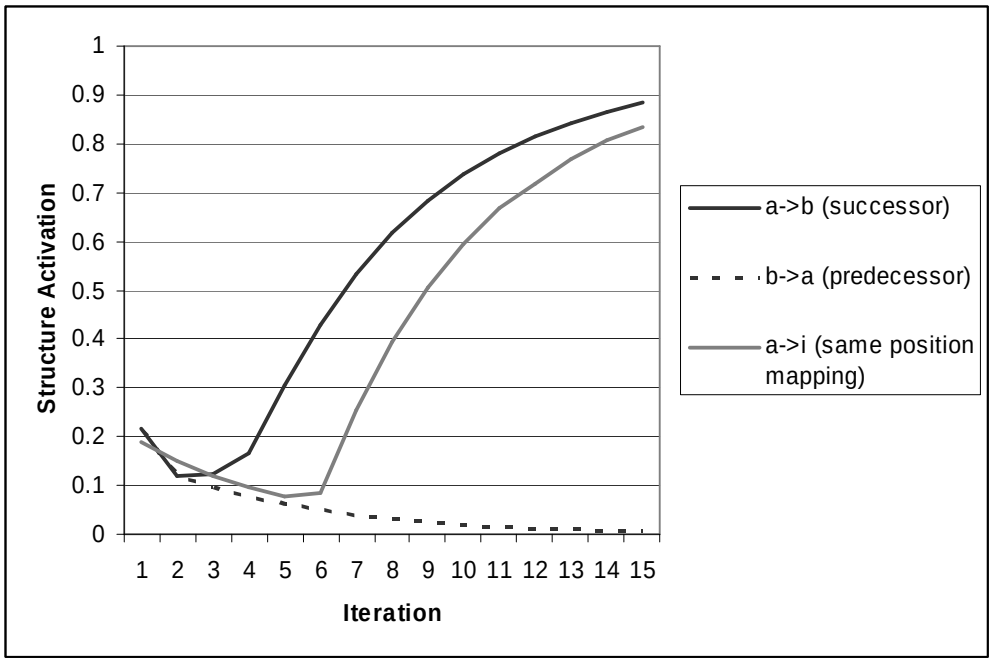


Figure 8.6 Creation of a consistent set of relations and mappings occurs through the resulting parallel constraint satisfaction network. In the implementation, the relationship between *a* and *b* can either be perceived as a right-going successor bond, or a left-going predecessor bond. However, as right-going bonds are preferred over left-going bonds (relating to a preference for scanning words in a left-to-right fashion in native English speakers), as well as there being a preference for noticing successor relationships, the successor bond between *a* and *b* gains the most activation over time, suppressing the alternative. As both the initial string (*abc*) and target string (*ijk*) will be perceived in this way, support is given to making correspondences between letters in the same rather than opposite locations (i.e., a→i, vs a→k).

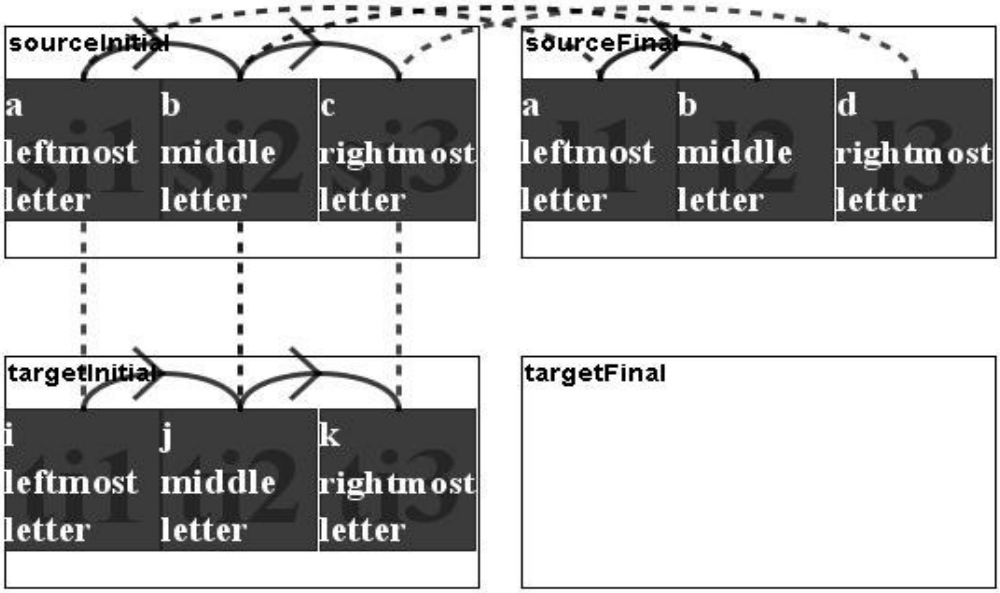


Figure 8.7 Over time, through facilitation and competition, a single consistent interpretation arises. In this case, both strings are perceived as containing sequences of successive letters to the right with letters in the same spatial position being perceived as playing the same role across strings.

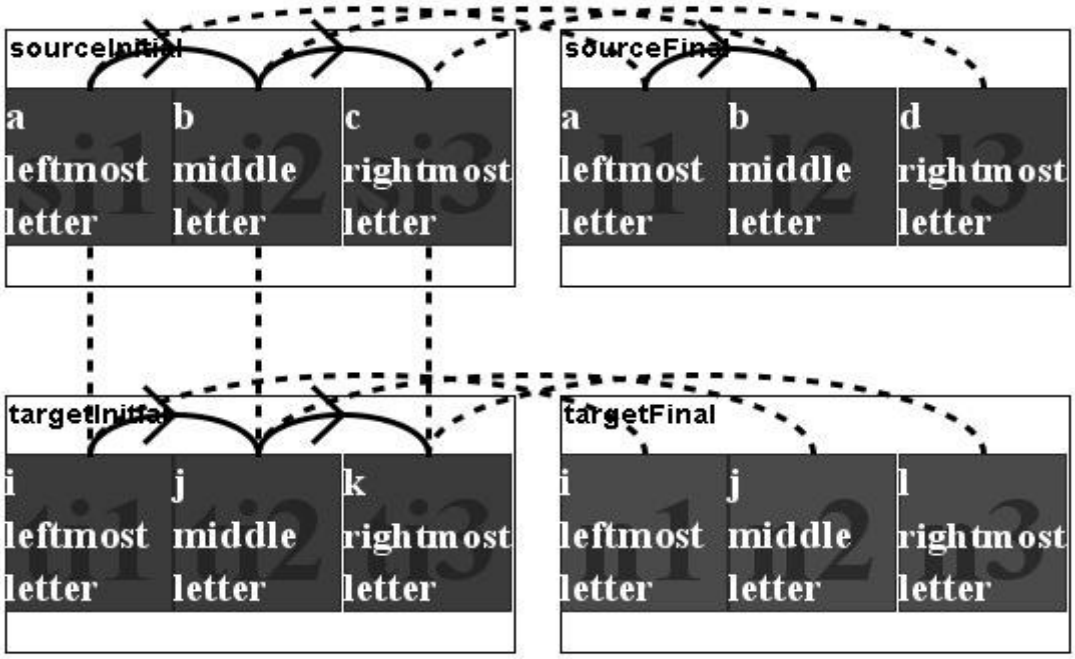


Figure 8.8 Based on the perceived relationships between the letters in the source initial and source final strings, information is "transferred" to the target domain in the formation of a solution string. In the above case, the first two letters in the source Initial and Final strings remain unchanged, being perceived to be linked through a "sameness" relationship. Thus, using the same relationship in the target domain, the letters "i" and "j" are proposed as the start to the solution string. In the above case as "c" is deemed to be related to "d" through the "successor" relation, the letter "k" is transformed in the same way, yielding the overall solution "ijl."

8.3.2 An alternative solution

The letter analogy domain is interesting in that human participants can generate a range of possible solutions to the same problem (Mitchell, 1993). In FAE, a range of answers can also be generated, as there are often several competing basins of attraction within the parallel constraint satisfaction network. For example, in the problem, abc:abd::ijk:?, the solution "ijd" can also be produced. As explained earlier, this solution arises when the relationship between "c" and "d" in the initial analog is perceived as "different" rather than "successor." As there is no unique object that is "different" to "k", the original information is transferred intact leading to the solution "ijd" (see figure 8.8).

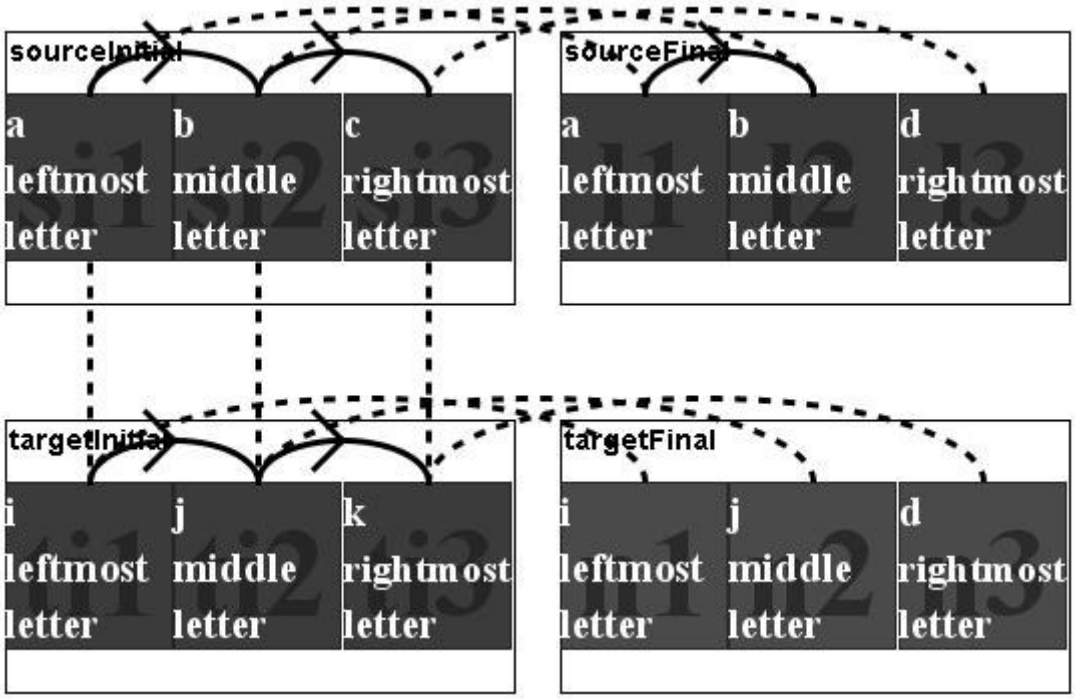


Figure 8.9 An alternative solution to the problem abc:abd::ijk:?.

8.3.3 Problem #2: abc:abd::iijjkk:?

The most common response generated by humans oo the problem *abc:abd::iijjkk:?* is *iijjll* (Mitchell, 1993). This solution requires the use of chunking to treat the pairs of identical letters as higher-order objects. Figures 8.9-8.15 describe the solution formation process in FAE in the generation of this solution.

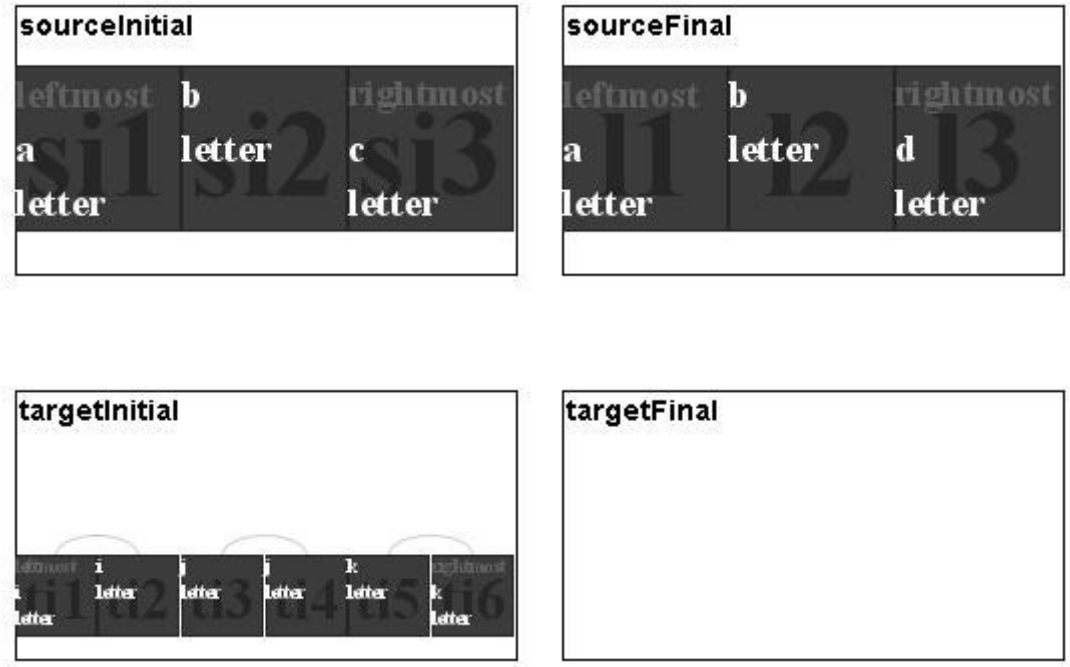


Figure 8.10. Given the raw problem descriptions, apart from property former productions noting the spatial positions of letters, bond former productions quickly note “sameness” relations between pairs of adjacent identical letters.

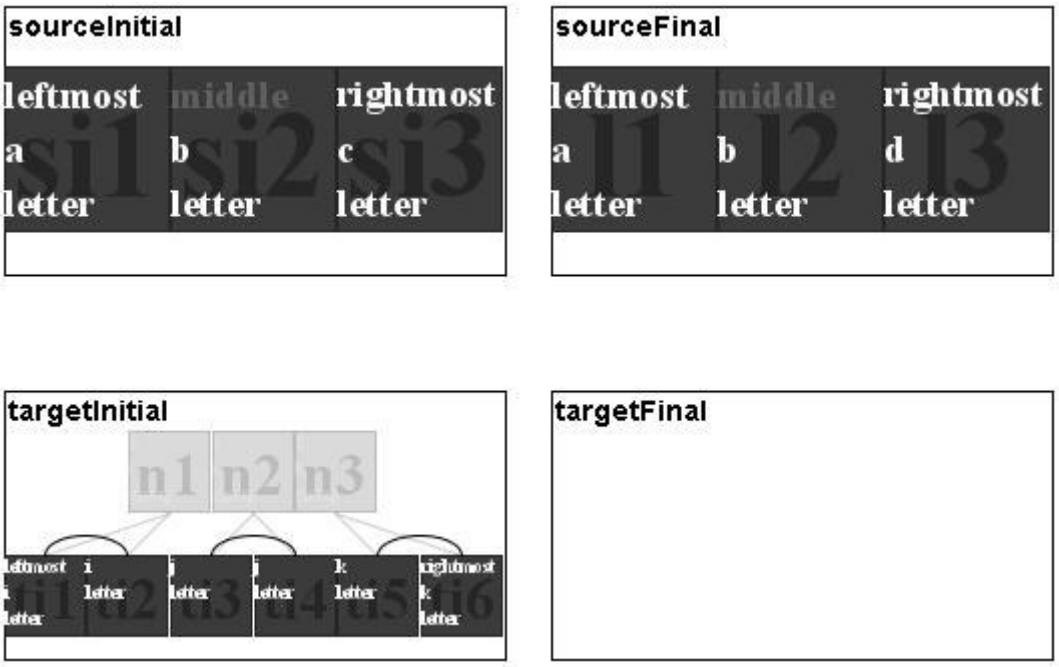


Figure 8.11. Based on the existence of "sameness" bonds, the three pairs of identical letters become chunked into higher order pairs.

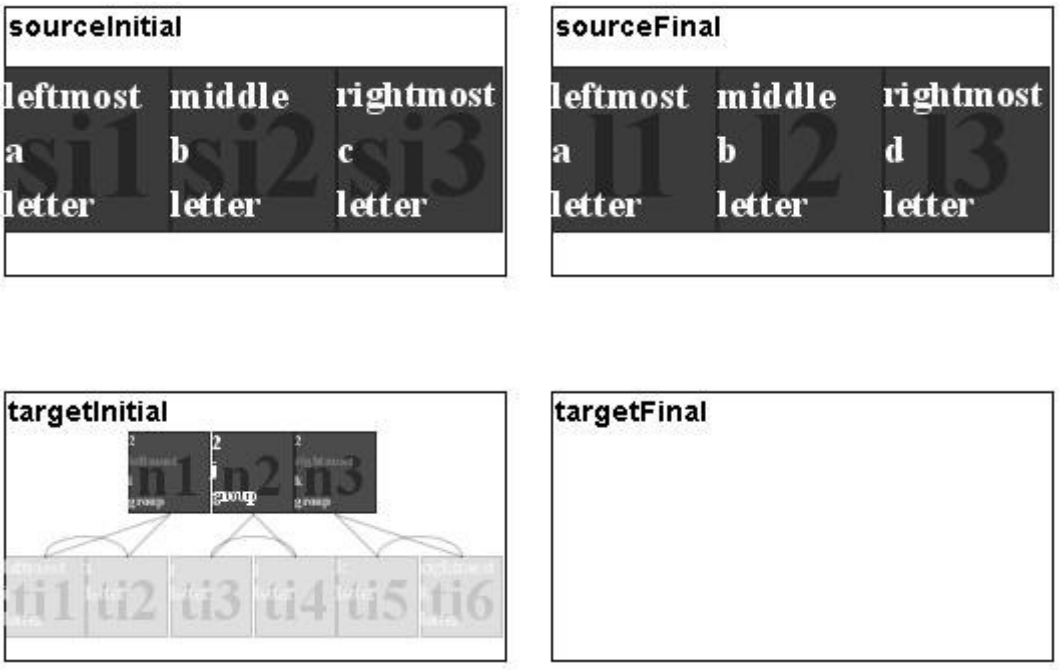


Figure 8.12 The new objects corresponding to chunked pairs of letters acquire their own set of properties, such as letter category and spatial location.

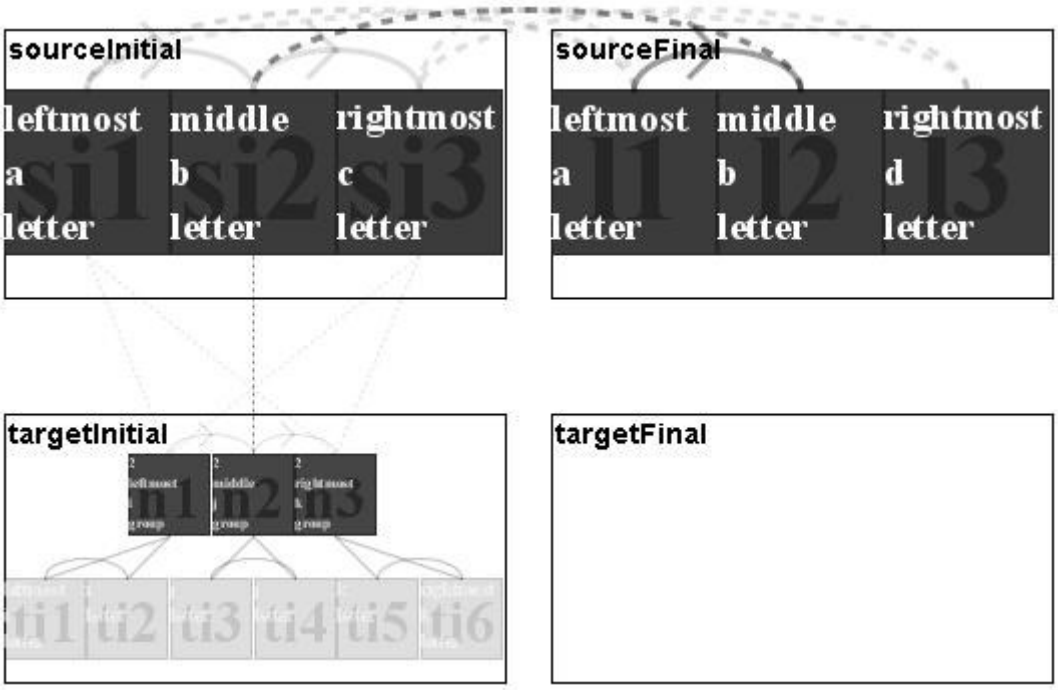


Figure 8.13 As with the previous problem, correspondences and bonds arise in parallel, in the formation of a consistent set of mappings.

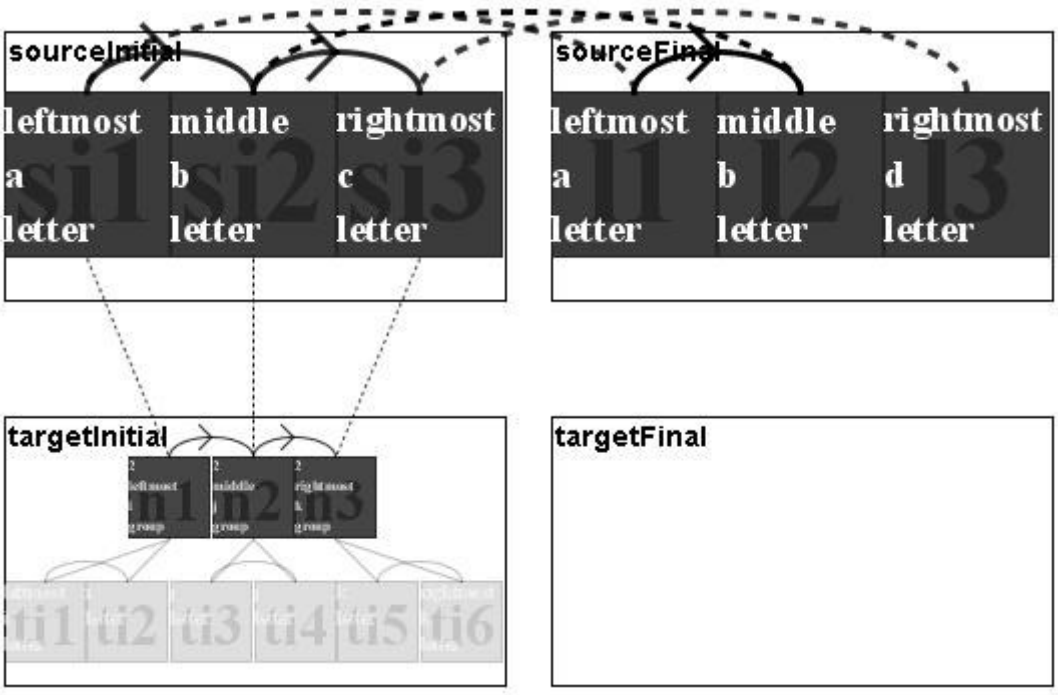


Figure 8.14 Over time, a single consistent set of mappings and bonds prevails. In the above example, the letter “c” is mapped to the “group” of two k’s in the target string.

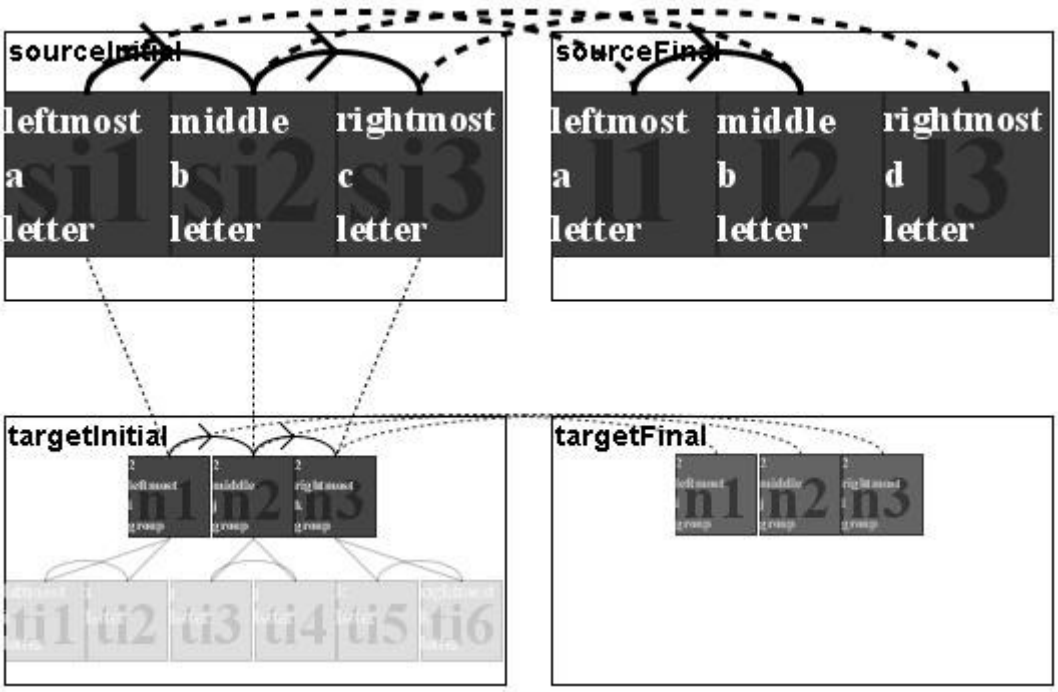


Figure 8.15 Once the mappings are resolved, the new information is transferred into the target domain, replacing the “k” descriptor by its successor.

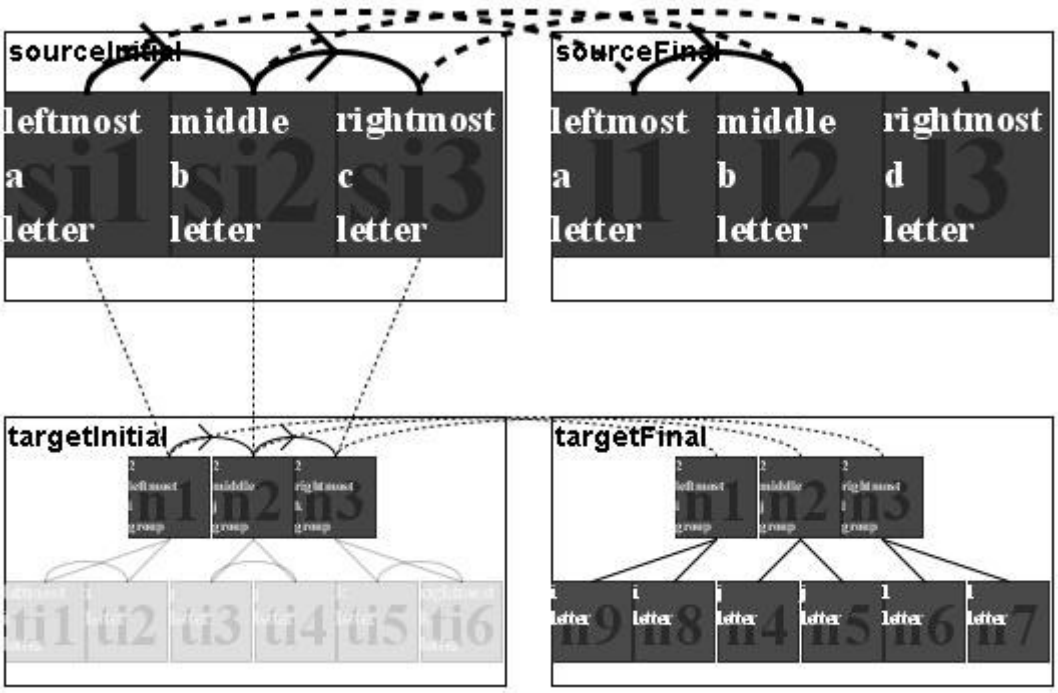


Figure 8.16 For consistency in this domain, the new target objects are “unpacked” from their chunked descriptions, to form the final solution string “iijjll.”

8.3.4 Problem #3: abc:abd::kji:?

This problem differs from the previous two in that common solutions given by humans include *lji* and *kjh* which involve conceptual slippages across the domain. Such slippages either exist in terms of mappings between different spatial positions (i.e. *c* being mapped to *k* in the formation of the solution *lji*), or in terms of slippages in the relation applied in transforming the target string (i.e. *kjh* requires that *i* is replaced by its predecessor rather that successor). Figure 8.16 to 8.20 illustrate the formation of these solutions.

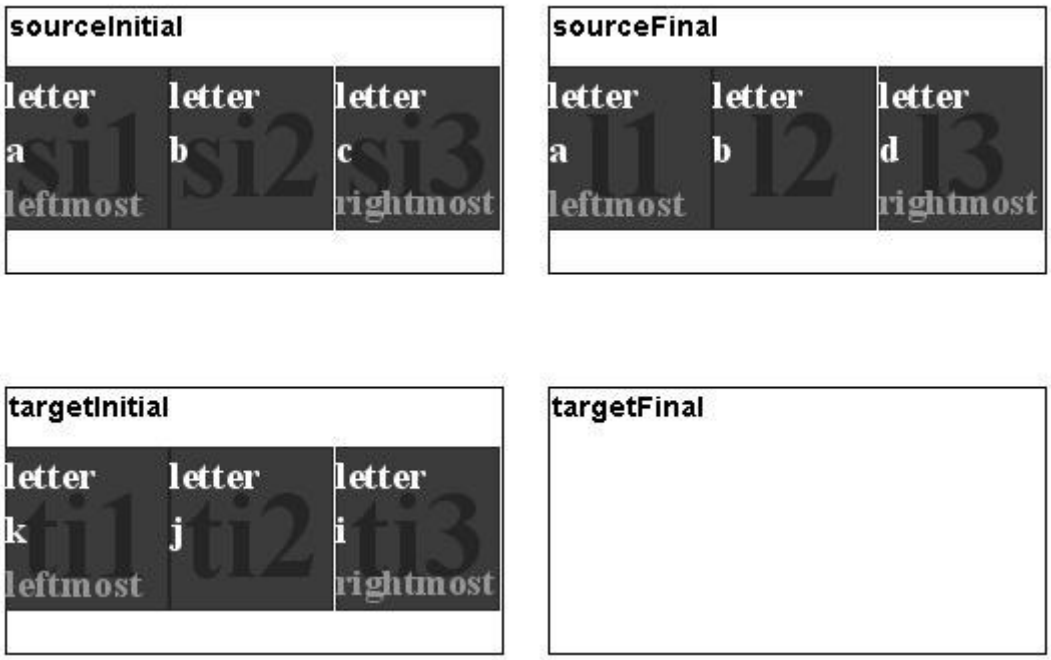


Figure 8.17 Given the initial state of the workspace, as in the previous problems, objects are assigned spatial positions.

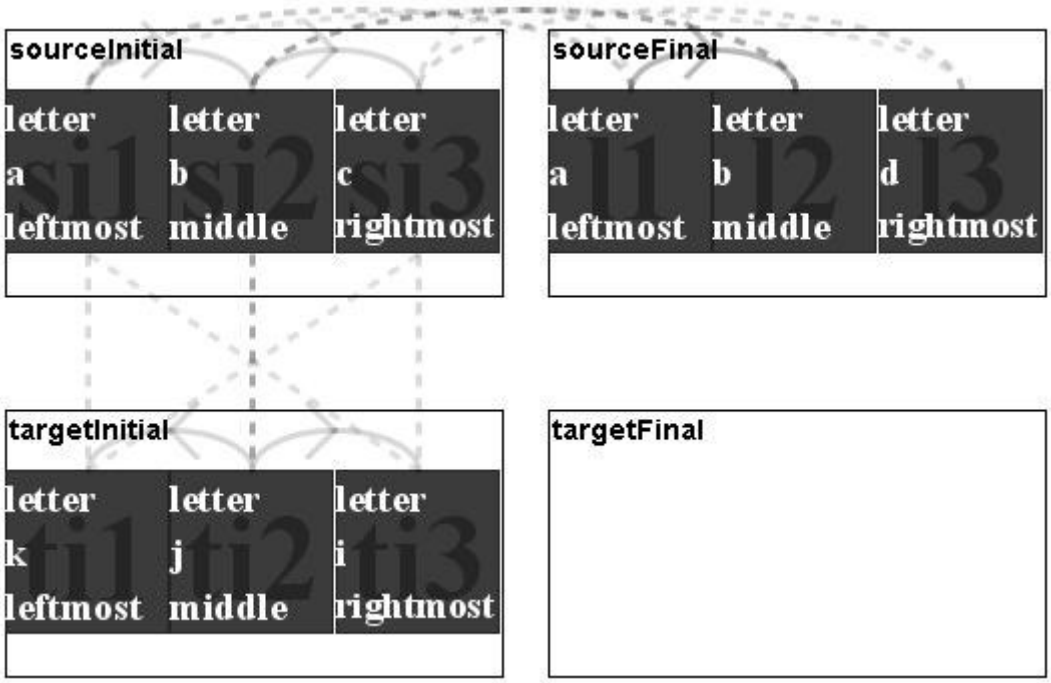


Figure 8.18 Due to the pressure to prefer the formation of right-going and successor bonds, although the initial source string is likely to contain right-going successor bonds, the target string is ambiguous (i.e. either being perceived as containing right-going predecessor bonds, or left-going successor bonds). In the latter case, the formation of these bonds supports the mapping of objects at opposite ends of the string.

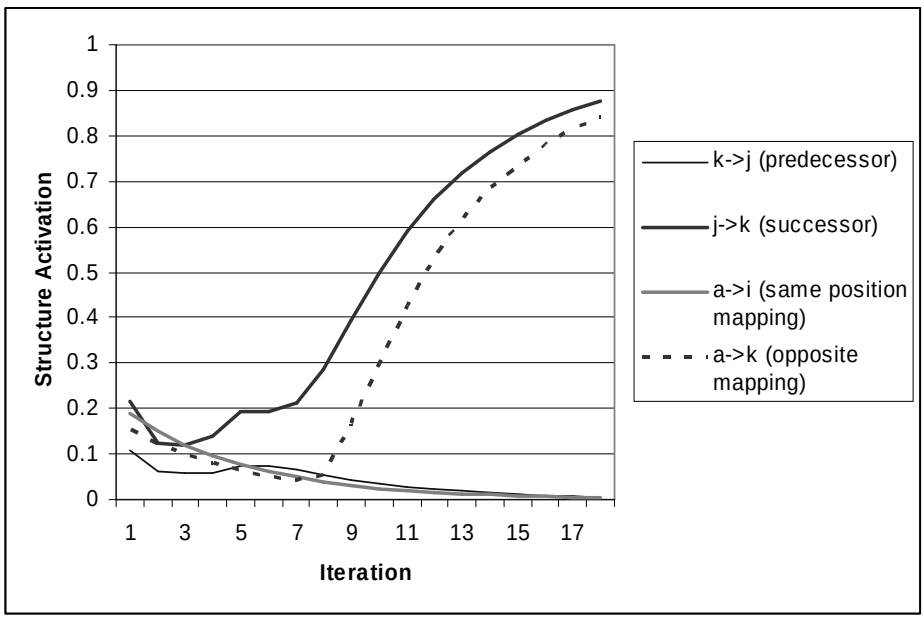


Figure 8.19 As there is a bias for creating right-going and successor bonds, the string *kj* is likely to be perceived as either containing a right-going predecessor bond, or a left-going successor bond. In the former case, mappings between letters at the same position will be supported, and in the latter, mappings between oppositely located letters will be supported. In the above case, the parallel constraint satisfaction network settles into a solution where the letters in the target string are perceived as being left-going successors, supporting mappings between oppositely located letters.

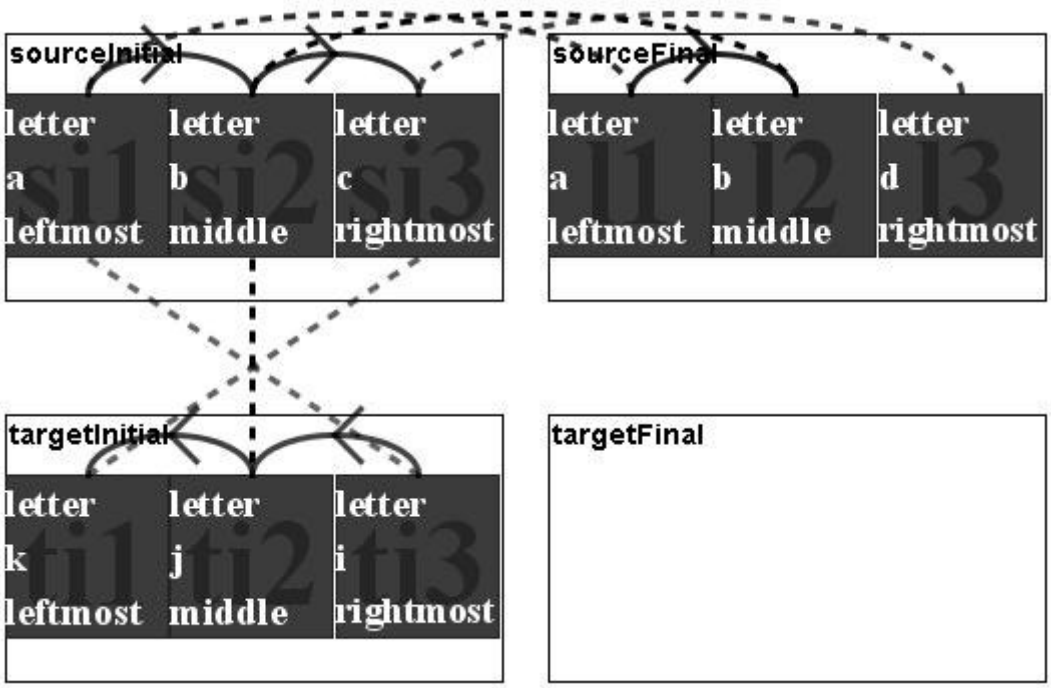


Figure 8.20 Over time, a single consistent set of mappings emerges, with objects at opposite ends of the string being mapped to each other. In this case, all bonds are "successors", being right-going in the source domain, and left-going in the target domain.

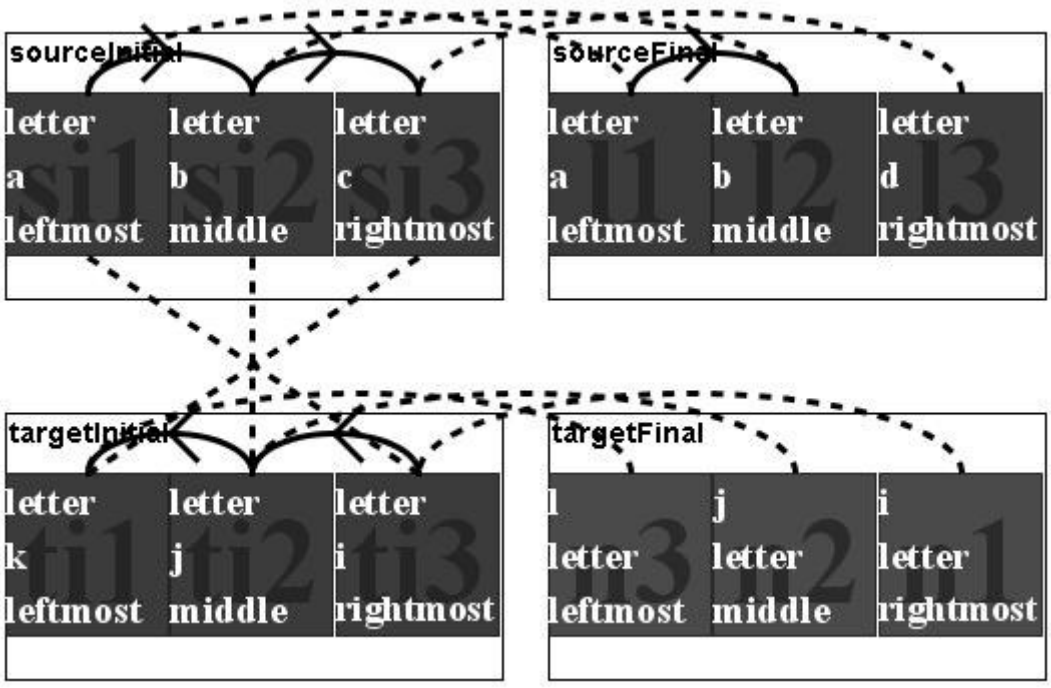


Figure 8.21 As the rightmost letter in the source domain, is mapped to the leftmost object in the target, it is the leftmost letter that will be altered in forming a solution. As there are no "conceptual slippages" in bond type, the leftmost letter will be replaced by its successor (as in the source analog), leading to the solution "lji."

8.3.5 An alternative solution

As stated, there can be several competing basins of attraction within the formed constraint satisfaction network, with the string “kjh” being an alternative solution to the problem abc:abd::kji:?. In this case, all objects at the same spatial positions are mapped across analogs and both strings are perceived in terms of right-going bonds. However, in the initial string, these bonds will be successors, whereas in the target they will be predecessors (leading to the conceptual slippage successor→predecessor). Applying this slippage to the target string transformation results in the rightmost letter being replaced by its predecessor rather than successor (as shown in figure 8.20).

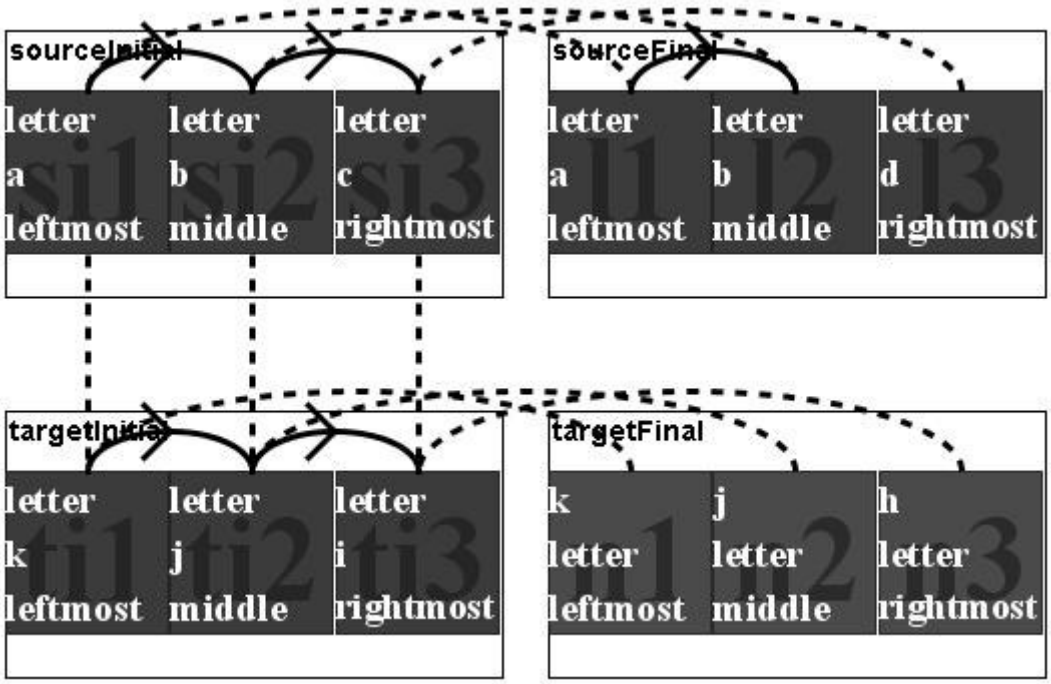


Figure 8.22 The mappings required for the solution “kjh” to occur.

8.4 Discussion

The current implementation solving letter-string analogy problems differs from the Copycat model (Mitchell, 1993) in two main ways. Firstly, in Copycat, the

phenomenon of conceptual slippage was modelled explicitly through the use of slip links within the semantic network that encoded potential conceptual slippages (such as the potential slippage between leftmost and rightmost). In order to make such slippages more or less likely to occur, the length of such links could vary over time, being a function of the activation of the node describing the relation (such as "opposite"). The regulating node (i.e. "opposite") however, only would become active when certain events within the workspace indicated that such a slippage was likely. For example, in the problem abc:abd::kji:?, the string *abc* could be interpreted as a right-going successor group, and the string *kji* could be interpreted as a left-going successor group. If a correspondence formed between these two groups (based on the fact that they are both successor groups), the opposite node would become activated (as the mapping includes a mapping between the concepts *right* and *left*). Once this node became active, mappings between objects at opposite end of the string were then more likely to form, leading to the solution *lji*. In the FAE implementation however, no such explicit means of encoding conceptual slippage was required. Rather, conceptual slippage was driven by local context, using the same form of parallel constraint satisfaction network that was used to model the word superiority effect (described in chapter five). Thus, in the FAE implementation, the "multiple activations" dilemma described in chapter two is avoided (i.e. the problem occurring when several instances of the same concept occur under different contexts), as there is no "single" global interpretation of likely slippages. The approach to creating mappings using a parallel constraint satisfaction network is consistent with other models of analogy-making such as AMBR (Kokinov, 1994a) and ACME (Holyoak and Thagard, 1989), that have been successfully applied to a range of analogical reasoning tasks outside the letter domain (all without the use global contextual definitions of concepts).

A second main aspect in which the current simulation differs from Copycat is in the way that information is transferred across domains. As described, in Copycat, this was achieved through the use of a single rule (such as "replace the leftmost letter by its successor"), whereas in the FAE implementation, this was achieved through the transference of any missing object that was linked to mapped objects. The approach taken by FAE is consistent with other models of analogy-making such as the Structure-Mapping Engine (Falkenhainer, Forbus & Gentner, 1989) and AMBR

(Kokinov, 1994a), that transfer information based on missing conceptual structures, rather that use of high-order rules. This approach is beneficial as is has been applied to a number of different domains, using the same underlying mechanisms. Specifying a higher order rule by contrast, seems domain dependent, requiring different forms of rule for different situations.

In the domain of letter-string analogy problems, the approach utilised by FAE in transferring information to the target does have its limitations. For example, it is assumed that each object in the source domain can only be mapped to a single object in the target domain. By contrast, Copycat allows for transformations such as "replace c by d", which will replace all instances of this letter in the target string. Furthermore, FAE is perhaps too precise in taking into account the conceptual slippages that occur between the domains. For example, in the problem *abc:abd::kji:?,* a prominent solution generated by people is *kjj*, which is not generated by FAE. Instead, in FAE, mapping objects in the same spatial locations will lead to the solution "kjh", as the initial string will be perceived as a group of successive letters, whereas the target string will be perceived as a group of predecessors (leading to the slippage successor→predecessor). In adapting FAE to generate *kjj*, it could be easily assumed that in transferring information between domains, such conceptual slippages are not always perceived, instead occasionally preferring the use of the original transformation. Although this is only a slight change to the system, an explicit and possibly empirically validated theory as to why such slippages do not always occur is required.

8.5 Summary

This chapter investigated the process of analogy-making in FAE. Using the same underlying mechanisms used for planning and flexible recognition described in previous chapters, FAE is able to adequately perform the same letter-string analogy problems as the Copycat model. The FAE model differs from Copycat however, in that no global definition of context was required, capturing all such "slippages" in terms of local contextual effects. Thus, the current approach addresses one of the main limitations of Copycat and Tabletop, namely the problem that arises when several instances of the same concept appear in different contexts at the same time.

FAE also differs from Copycat in the use of a parallel constraint satisfaction network and a domain-independent method of knowledge "transfer" during solution formation. These mechanisms have proven fairly general, as are employed by other models of analogy-making that can function in a range of different domains.

Chapter 9
General Discussion

The Fluid Analogies Engine has been developed as a new computational framework for modelling flexible and intelligent behaviour that is capable of capturing both high and low levels of cognitive processing (e.g., "mental deliberation" and "perception"). Encompassing both these levels within a single framework is important as not only are both levels central to intelligent behaviour (i.e. there are many "direct" pathways between perception and action in which "mental deliberation" is not required), but it is evident that these processes are heavily co-dependent (e.g., the way a situation is perceived is often determined by the task being performed). As a more concrete guideline to evaluating such a system, six abilities that are central to flexible and intelligent cognition that span these two levels were identified. These abilities include:

1. *The ability to process raw distributed information*
2. *The ability to select and attend to relevant aspects of the world*
3. *The ability to process information in a contextually sensitive manner*
4. *The ability to create and manipulate complex hierarchical representations*
5. *The ability to exhibit rational, goal-directed behaviour*
6. *The ability to perform self-watching and mental regulation*

Although approaches to cognitive modelling capture subsets of the above properties, there are no current systems that are well suited at performing all of these abilities (as discussed in chapters two and three). This thesis has described a new hybrid cognitive architecture called FAE (the Fluid Analogies Engine) aimed at addressing this gap in current research. FAE integrates an array of ideas from the different modelling paradigms into a single system that is capable of capturing both high and low level processing tasks and displaying all six of the mentioned cognitive abilities underlying

flexible cognition. In Chapters five to eight, simulations testing the capacity of FAE to display these abilities in a range of tasks were described.

9.1 Summary of Results

In this thesis, simulations testing FAE's ability to function in a set of four different domains were described. Rather than attempting to model human performance at a fine-grained behavioural level (i.e. trying to match exact response frequencies), the simulations examine coarse-grained human competencies (or abilities) that as a whole, have not been modelled within a single framework. As a testbed, the four simulations when taken together, require a system to exhibit all six of the cognitive abilities mentioned in chapter 1. The following sections summarise and discuss FAE's ability to display these general cognitive skills.

1. The ability to process raw distributed information

All processing in FAE is achieved at a subsymbolic level, utilising LEABRA's update and transfer functions. In these models, all information is represented in terms of distributed vectors that share information through weighted excitatory links. Levels of activation change gradually over time, and are regulated through a normalisation procedure that restricts the amount of activation that can occur within each vector (implemented by a fuzzy k-winner-take-all algorithm). Reusing the LEABRA algorithms in FAE is important in that they have proven well equipped at handling raw distributed information, having been used to successfully model a wide range of low-level cognitive phenomena. Example simulations include the processing of natural scenes (learning weight patterns mimicking the representations found in the primary visual cortex), learning to pronounce nearly 3,000 English words (generalising to novel nonwords), and learning the semantic features of words embedded in raw text (O'Reilly, 2000).

All the simulations described in this thesis have demonstrated FAE's ability to process distributed information, as all concepts known to the system were encoded in these terms. The ability to "process raw distributed information" however, was

perhaps most demonstrated in the word superiority effect simulations in which the input was in the form of overlapping distributed visual features. In processing this information, particular patterns (i.e. letters) were detected in a graded fashion by nodes in the "letter" layer. In this system, noisy and ambiguous letters that were placed in the context of words could be identified correctly, through an interaction between bottom-up and top-down processing. Thus, this simple simulation demonstrates FAE's ability to process information presented in the form of raw distributed information, using the representations and algorithms of LEABRA.

Although in many of the simulations described in this thesis each concept was defined using a local encoding scheme in which only a single bit was on, the action of any single production often manipulated several such concepts simultaneously. The result of such productions was the creation of attractor basins involving sparse binary representations, in which several bits were active. For example, in the Jumble domain, the creation of a consonant cluster involved the creation of a new object with two property values and two relations, resulting in an appended vector representation in which four bits were active (i.e. one vector for each property and relation). Thus, all the simulations described in this thesis demonstrate at some level FAE's ability to manipulate sparse distributed vectors, whether it be in representing low-level sensory data or high-level symbolic structures.

2. The ability to select and attend to relevant aspects of the world

According to the connectionist paradigm, the processes of selection and attention can be captured intrinsically in networks in which there are a competition for resources (O'Reilly, 2000). In FAE, such a competition is formed through two means: utilising LEABRA's k-winner-take-all algorithms (that restrict the number of highly active units), and through the use of inhibition between incompatible objects and relations. In FAE, attention and selection were clearly visible in the simulations in the Numble and Jumble domains in which there were several alternative solutions, but with only a single above-threshold viewpoint being explored at a time.

3. The ability to process information in a contextually sensitive manner

In both the word superiority effect and analogy-making simulations, FAE's ability to process information in a contextually sensitive manner was demonstrated. That is, in the word superiority effect simulations, an ambiguous letter embedded in two different word contexts was recognised differently (but appropriately), and in the letter-string analogy problems, mappings between objects were determined by similarities in their context rather than surface similarities. In both of these situations, contextually dependent processing was achieved through the connectionist aspects of FAE that constructed a parallel constraint satisfaction network. Processing within this network was gradual and interactive, being consistent with the properties outlined by McClelland and Rumelhart (1981) as being central to accounting for this phenomenon in general.

4. The ability to create and manipulate complex hierarchical representations

In three out of the four domains explored in this thesis (namely the domains modelled by Jumbo, Numbo and Copycat), FAE was required to manipulate complex hierarchical representations. In processing such information, FAE utilised a dynamically constructed network that could accommodate a variable number of objects and relations. In this method, each separate object, relation and property was represented by a separate vector, with information flowing between such structures through intermediate hidden nodes. In accordance with traditional unified theories such as SOAR and ACT-R, transformations were described in terms of symbolic productions that contained variable bindings. In FAE, the resulting production instantiations were turned into hidden nodes that transferred information in a gradual and interactive manner. The resulting approach taken by FAE was capable of manipulating complex symbol structures like other models of high-level perception, but was able to do so in a flexible and context-sensitive manner.

5. The ability to exhibit rational, goal-directed behaviour

In both the Jumble, Numble and analogy-making domains explored in this project, there were specific goals that needed to be achieved (i.e. the creation of a word-candidate, the formation of the target number from a set of smaller numbers, and the formation of a letter-string solutions). In FAE, the formation of such solutions "emerged" through the parallel actions of many sub-processes that, as a whole, explored the state space in a rational and goal-directed manner. For example, in the Jumble domain, common letter sequences had a higher probability of forming, biasing the search to try these alternatives first. Although the combination of processes did not always lead to a direct answer (i.e. backtracking was frequently required), the combined processes would drive the search to explore likely avenues, heuristically reducing the search space.

6. The ability to perform self-watching and mental regulation

During exploration of the domains of Jumble and Numble, the heuristics employed by FAE in driving the search would often lead the system down a dead-end path. FAE could backtrack out of such impasses through the use of its episodic memory that inhibited structures with a strong memory trace. That is, when an impasse state was reached, the episodic trace for the state (the set of elements in working memory) would increase until a threshold was surpassed. At this point, such structures would be inhibited and avoided in future explorations. Thus, FAE contained simple but effective mechanisms for self-watching (i.e. an episodic memory for working memory states), and mental regulation (i.e. the inhibition of states with strong episodic traces).

9.2 Comparisons to Previous Models

The main distinction between FAE and the other models of high-level perception discussed in this thesis (e.g., ACT-R, SOAR, and the various FARG models), is that in FAE, a major attempt has been made to bridge the gap between symbolic and subsymbolic computation. That is, in FAE "symbolic computation" arises through subsymbolic processing, involving the same form of attractor basins and distributed

vectors that afford low level perceptual processing. Such a unification of processing levels is essential in capturing the flexibility of human cognition. For example, although much of high-level reasoning can be classified as symbolic, both the symbols and "rules" for manipulating them need to be perceptually grounded. For example, in reasoning about how to gain the appropriate height to change a light bulb, standing on a chair may be a typical solution. However, the choice of chair is evidently governed by many subsymbolic decisions, such as its weight-bearing characteristics, height, and balance (e.g., a rocking chair may not be a good option). One of the main features of the FAE framework is that it explores mechanisms for the "perceptual grounding" of such symbols and rules.

In integrating symbolic and subsymbolic processing, FAE has several important features that distinguish it from other prominent models of high-level cognition such as ACT-R, SOAR and the FARG models. Firstly, in FAE, all concepts are represented as (potentially overlapping) feature vectors, affording a natural grounding to perception. This contrasts the other models that represent concepts as discrete entities. Although in such systems, the global similarity between two concepts can be represented in terms of a weighted "similarity" link, the subtle information as to "how" they are similar (which is often essential to reasoning) is lost.

The other major distinction of FAE from other prominent model of high-level cognition is that information flows through FAE using parallel connectionist algorithms, affording local competition and interaction. As explained in chapter 5, such interactions are useful in explaining contextual dependencies in processing at both low and high levels of perception. Such an approach differs from SOAR and ACT-R, in which productions fire in an all-or-nothing fashion, not providing mechanisms for local contextual dependencies. Although in the FARG models Codelets fire in parallel, gradually updating the "proposal" level of the symbolic structures, they do not provide local facilitation and competition, instead relying on activation from a semantic network to set a global context. Thus, FAE is unique in that it proposes global mechanisms that are appropriate for contextually dependent processing at both low and high levels of perception.

Out of the various alternative cognitive architectures reviewed in this thesis, FAE's closest relatives are the more complex of the FARG models, including Copycat (Mitchell, 1993), Tabletop (French, 1995) and Metacat (Marshall, 1999). Apart from the differences mentioned above, FAE differs from these approaches in a number of other important ways. In these FARG models, the Slipnet (the equivalent of FAE's long-term memory), had a crucial role in solution formation, firstly, by spawning Codelets that actively seek for instances of the active concepts (requiring Codelets that correspond to each concept), and secondly, by modifying the "similarity" links to make conceptual slippages more or less likely (which is important in mapping in analogy-making). In the first instance, FAE differs from these models in that none of the concepts in long-term memory actively trigger productions. Instead, in FAE, concepts are retrieved from memory by productions, given a set of cues. This approach allows for more general productions to be written, affording better scalability than the FARG models. For example, if modelling the ability to count, only a single production would need to be written, retrieving from memory the successor of the current number in working memory. In the FARG models, it is hard to tell how the equivalent process could be implemented.

With respect to the second role of the Slipnet in the FARG models mentioned above (i.e., implementing contextual sensitivity), equivalent processes exist within FAE, but at a different location. Instead of modifying concepts at a conceptual level with changes in context, conceptual sensitivity in FAE occurs through the interactions of the various productions. For example, in Chapter 5, an ambiguous letter was perceived to be a P in the word "HEL*" and an A in the word "RE*D". This was achieved through top-down activation of likely word candidates that provided a context for disambiguation. In such cases, a modification of conceptual boundaries is not required within long-term memory. Also, allowing context to be defined locally rather than globally, allows FAE to avoid the "multiple activations" dilemma noted by French (1995) that occurs when different instances of the same concept occur in different contexts in working memory (discussed in section 2.3.4). Thus, FAE provides a more general mechanism for contextual sensitivity than that proposed in the FARG models.

Although it could be argued that cognitive architectures such as SOAR are very general and could be programmed to address all the limitations noted above, the major difference between these approaches lie in the forms of processing that they naturally afford. Context-sensitivity and the ability to process raw distributed information are two features central to human cognition are naturally accommodated for in FAE, but would require a great deal of effort to be modelled using SOAR, ACT-R or the FARG approach.

9.3 On the Creation of a Modelling Framework Exploring both High and Low Level Perception

In designing a cognitive modelling framework that equally handles both high and low level processing, several important issues are raised. Firstly, as there is no clear-cut boundary between these levels (i.e. perception and reasoning are intimately tied), representations that are applicable to both levels need to be specified. As distributed vectors are almost synonymous with low-level processing, the real question raised by this thesis was how the complex hierarchical structures required for high-level reasoning and planning could be described in such terms. Although there have been connectionist models proposed such as the *Recurrent Auto-Associative Memory* (RAAM) that can encode symbolic tree-like structures (Pollack, 1990), it is not clear how appropriate these systems would be in representing working memory. That is, in tasks such as Jumble, a variable number of objects need to be stored, but with all objects needing to be being equally visible to perception and manipulation. It is not evident that using a network such as RAAM would provide such easily accessible representations. A second alternative would be to use a fixed number of spiking neurons that are capable of representing a variable number of objects and their associations through a temporal sequence of firing. Again, it is not clear how such a representational scheme could interface easily with procedural memory. That is, it is not obvious how a production-like rule can be triggered by a temporal sequence of events, and how the resulting action can affect the firing rates of neurons to create or destroy structures in the sequence. In FAE, a hybrid approach has been employed to resolve this issue, allowing the system to dynamically allocate new vectors for each unique object, relation and property. Although not biologically plausible, the

approach represents the same kind of structures that would be required to be embedded in a temporal sequence in a spiking network (i.e. turning the temporal encoding into a spatial one). Further investigation is required to evaluate the relative merits of the various approaches.

A second question raised in the implementation of a model of high-level perception utilising lower-level representations, is how invariant rules can be described and implemented. For example, in the Jumble domain, letters needed to be grouped into frequently occurring sequences irrespective of their original position in the string. For example, given the letter sequence *t-r-n-i-g-s* and a rule for proposing that the letters "s-t" make a good combination, the system needs to be able to identify the letters "s" and "t" irrespective of initial location. How such invariances can be determined is strongly related to the way in which information is represented in working memory. If this information is encoded using a spiking network for example, the issue becomes how to identify the required structures irrespective of how far apart they are temporally. In FAE, in contrast, the issue becomes how to perform spatial invariance, as each unique property is represented by a different vector. In FAE, this issue is addressed using a production system that allows for variable binding. That is, a single rule can be proposed, with the system checking all combinations of objects for the specified pattern. This is consistent with most prominent models of high-level cognition such as SOAR, ACT-R and the Fluid Analogy models.

A third major question that is raised in the implementation of such systems is in the general architectural components that are necessary for intelligent cognition and how they interact. FAE is based on the general architecture of the Fluid Analogy models, incorporating a working memory, semantic memory, procedural memory and episodic memory. In these models, it is assumed that all processing occurs in parallel, including retrievals from semantic memory. This assertion conflicts with models such as ACT-R that assume that only one "fact" can be retrieved at a time from memory. Without such parallel retrievals however, in tasks such as analogy-making, representation formation would become more of a serial rather than parallel task. The alternative would be to suppose that the processes such as building "successor bonds" are more automatic than suggested by models such as Copycat, being carried out by a large set of different productions without the need for explicit retrieval. Again,

further work is required in evaluating the merits (both in terms of the practicality and psychological validity) of the various alternatives.

9.4 Implications for Theories and Models Bridging the Gap between Low and High Level Perception

In attempting to bridge the gap between perception and reasoning, many hybrid approaches have been proposed that merge symbolic manipulation with subsymbolic processing. Such models include ACT-R, 3Caps, DUAL and the various Fluid Analogy Models. In all of these models, a top-down approach has been taken, in which symbolic computation has been augmented with a subsymbolic component, typically in the form of the graded activation levels of concepts and the relative speed at which symbolic computations occur. Such resulting approaches however, have a major omission in that they say little about the mechanisms required for low-level perception (i.e. the processing of raw distributed information).

The theory underpinning the development of FAE is markedly different from that mentioned above. Rather than taking a top-down approach, FAE takes a bottom-up approach, asking how the same mechanisms that afford the processing of raw distributed information can be applied to high-level perception. Although the same question has been raised by the connectionist paradigm, connectionist models have not yet proven as powerful as the symbolic approach at modelling high-level tasks as planning and reasoning. In overcoming the inherent limitations of connectionism, FAE takes a hybrid approach, exploring the augmentations required to the connectionist paradigm in order to effectively process complex symbol structures.

The development of FAE has demonstrated that the basic mechanisms of connectionism required for low-level processing (i.e. an interactive and gradual flow of information between distributed vectors) can effectively be utilised in a hybrid system to display the same levels of symbolic processing as prominent models of high-level cognition. In this system, rather that having explicit procedures for selecting the next action to perform (as in the selection of the production with the highest “utility value” in ACT-R), action selection in FAE is handled implicitly

through the use of attractor basins within the constructed network. These attractors also implicitly model the flexible and context-sensitive nature of human perception, with contextual factors supplying extra input that can bias which stable state the system will fall in to. The general approach taken by FAE differs from models such as Copycat in which potential conceptual slippages were modelled explicitly through links in a semantic network. In contrast, FAE models conceptual slippage as a natural consequence of the interaction between distributed vectors (that implicitly represent the similarity between concepts) and contextually sensitive basins of attraction.

9.5 Conclusion

In modelling human cognition, different levels of processing (i.e. low-level "automated" processing verses high-level "symbolic" processing) have typically been modelled using disparate approaches (e.g., connectionism versus the traditional symbolic approach). However, in many examples of human reasoning (such as choosing an appropriate object on which to stand to change a light bulb) it is apparent that these levels are inextricably tied. That is, symbols and rules in the mind cannot be viewed as rigid all-or-nothing entities, but rather, have fuzzy, context-sensitive boundaries that are grounded in perceptual representations. Thus, in order to effectively model flexible high-level cognition, frameworks need to be developed that bridge the gap between low-level subsymbolic processing and high-level symbolic processing. In this thesis, a new cognitive modelling framework (the Fluid Analogies Engine) was described that has made substantial progress in bridging this gap.

The Fluid Analogies Engine offers a new framework for cognitive modelling that combines the strengths of the connectionist and symbolic approaches in order to capture both high and low levels of processing. FAE has demonstrated that the same basic mechanisms necessary for low-level processing (i.e. the facilitation and competition between distributed representations), can also be effectively applied to high-level perception within a hybrid system. That is, the use of attractors implicitly affords conflict resolution and the modelling of flexible context-dependent concepts; properties that are captured by explicit mechanisms in "symbolic" models such as ACT-R and Copycat. Thus, FAE offers a new unifying framework for modelling both

low-level and high-level cognitive processes (e.g. perception and reasoning), using common representations and processes that afford a seamless integration between these levels of processing.

In modelling human cognition, the use of "attractors" to account for both high and low levels of cognition is only part of a solution. In order to capture core human competencies (such as the ability to process raw distributed information as well as representing and manipulating complex symbol structures), a non-unitary architecture is useful, including such elements as low-level processing layers, a working memory, a long-term declarative memory, and a procedural memory. Apart from including such components, FAE specifies a unique set of processing mechanisms that afford communication between these subsystems and allow the system as a whole to capture the core set of six cognitive abilities outlined in chapter one. In particular, in FAE, all information is represented through dynamically allocated vectors that allow for the representation of both raw distributed information as well as complex relational structures (with each object, relation and property being described by a different vector). Allowing the manipulation of such information, is a production system at the heart of FAE that dynamically constructs an attractor neural network affording subsymbolic transformations, conflict resolution, and the modelling of flexible context-sensitive concepts. Such a dynamic system allows for the flexible manipulation of complex (and potentially grounded) symbol structures: a central characteristic of human reasoning that is not well modelled by other prominent models of high-level perception such as SOAR, ACT-R or the various FARG models.

9.6 Further Work

One of the main limitations of the Fluid Analogies Engine is that all information known to the system is required to be hand-coded by humans. Apart from being impractical for large scale real-world environments, such representations do not often reflect the subtleties inherent to human knowledge that are required for flexible cognition. For example, the simple act of recognizing a face or "simple" acts of reasoning (such as choosing an object on which to stand in order to change a light bulb) require flexible knowledge about spatial and featural regularities not easily captured in terms of symbolic hand-coded rules. Such a wealth of information is

unlikely to be provided through explicit descriptions from human programmers, but rather, is only likely to be extracted through experience. Thus, future work is required to explore techniques that would allow FAE to learn its representations through embodied interactions with the world. Due to the importance of learning in cognitive systems, an effort has been made in the implementation of FAE to choose methods of processing that could support learning in future implementations. For example, in FAE, concepts are stored in long-term memory as a rank-three tensor (which can easily afford the learning of new information, given appropriate normalisation), and productions are implemented as a dynamically constructed neural network. In fine-tuning or acquiring new productions, it is expected that the learning algorithms employed by LEABRA (O'Reilly and Munakata, 2000) may offer important insights, as the connectionist aspects of FAE (i.e. the representations and transfer functions utilised) are consistent with this framework.

References

Anderson, J. R. (1983). *The architecture of cognition*. Harvard University Press, Cambridge, MA.

Anderson, J. R. (1990). *The Adaptive Character of Thought*. Erlbaum, Hillsdale, NJ.

Anderson, J. R. (1993). *Rules of the Mind*. Erlbaum, Hillsdale, NJ.

Anderson, J.R., & Lebiere, C. (1998). *The atomic components of thought*. Lawrence Erlbaum associates, Mahwah, NJ.

Anderson, J. R., Bothell, D., Lebiere, C. & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, 38:341-380.

Biederman, I. (1972). Perceiving real-world scenes. *Science*, 177(4043):77-80.

Boden, M. A. (1994). Agents and Creativity. *Communications of the ACM*. 37(7):117-121.

Bongard, M. (1970). *Pattern Recognition*. Spartan Books, New York.

Broadbent (1958), *Perception and Communication*. Pergamon Press, London.

Brooks, R. A. (1990). Elephants Don't Play Chess. *Robotics and Autonomous Systems*, 6:3-15.

Cherry, E. C. (1953) Some experiments on the recognition of speech with one and with two ears. *Journal of the Acoustical Society of America*. 25:975-979.

Deacon, T.W. (1997). *The Symbolic Species: The Co-Evolution of Language and the Brain*. W. W. Norton & Co, New York & London.

Falkenhainer, B., Forbus, K. D. & Gentner, D. (1990). The Structure-Mapping Engine. *Artificial Intelligence*, 41(1):1-63.

Flemming, U. (1987). The Role of Shape Grammars in the Analysis and Creation of Designs. In Yehuda E., editor, *Computability of Design*, pages 245-272. John Wiley & Sons, New York.

Franklin, S., (2001) *Artificial Minds*. MIT Press, Cambridge, MA.

French, R. M., (1995) *The subtlety of sameness : a theory and computer model of analogy-making.* MIT Press, Cambridge, MA.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193-202.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155-170.

Godden, D. R., & Baddeley, A. D. (1975). Context-dependent memory in two natural environments: Land and underwater. *British Journal of Psychology*, 66:325-331.

Godden, D.R., & Baddeley, A. D. (1980). When does context influence recognition memory? *British Journal of Psychology*, 71:99-104.

Halford, G.S., Maybery, M.T., O'Hare, A.W., & Grant, P. (1994). The development of memory and processing capacity. *Child Development*. 65(5):1338-1356.

Hill, W. E. (1915) "My Wife and My Mother-in-Law." *Puck*, 16(11).

Hinton, G.E., & Sejnowski, T. J (1983). Optimal Perceptual Inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washinton DC.

Hofstadter, D. R. (1979). *Gödel, Esher, Bach: An eternal golden braid.* Basic Books, New York.

Hofstadter, D. R. (1983). The architecture of Jumbo. *Proceedings of the International Machine Learning Workshop*. Monticello, Ill.

Hofstadter, D. R. & The Fluid Analogies Research Group (1995). *Fluid Concepts and Creative Analogies*. Basic Books, New York.

Holyoak, K. & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, 13(3): 295-355.

Just, M. A., Carpenter, P. A. & Hemphill, D. D. (1996). Constraints on processing capacity: Architectural or implementational. In D. M. Steier & T. M, Mitchell, (Eds.) *Mind matters: A tribute to Allen Newell*, pages 141-178. Erlbaum, Mahwah, NJ.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983) Optimization by Simulated Annealing. *Science*, 20:671-680.

Kirsch, J. L. & Kirsch, R. A. (1986). The structure of paintings: formal grammar and design. *Environment and Planning B: Planning and Design*, 13:163-176.

Kokinov, B (1989). About Modeling Some Aspects of Human Memory. In: Klix, F., Streitz, N., Waern, Y., & Wandke, H., editors, *Man-computer interaction research MACINTER-II*, pages 349-359. Elsevier Science Publishing, Amsterdam.

Kokinov, B. (1992). Inference evaluation in deductive, inductive and analogical reasoning. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pages 903-908. Erlbaum, Hillsdale, NJ.

Kokinov, B. (1992). Similarity in analogical reasoning. In Sgurev, V. & du Boulay, B., editors, *Artificial intelligence III: Methodology, systems, applications*. Elsevier Science Publishing, Amsterdam.

Kokinov, B. (1994a). A Hybrid Model of Reasoning by Analogy. Chapter 5. In K. Holyoak & J. Barnden (eds.) *Analogical Connections, Advances in Connectionist and Neural Computation Theory*, vol. 2, Ablex Publ. Corp.

Kokinov, B. (1994b). The DUAL cognitive architecture: A hybrid multi-agent approach. In A. Cohn (Ed.), *Proceedings of the Eleventh European Conference on Artificial Intelligence*. John Wiley & Sons, London.

Kokiov, B. (1994c). The context-sensitive cognitive architecture DUAL. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*. Erlbaum, Hillsdale, NJ.

Kokinov, B. and French, R. M. (2003). Computational Models of Analogy-making. In Nadel, L. (Ed.) *Encyclopedia of Cognitive Science.* 1:113 - 118. Nature Publishing Group, London.

Kokinov, B. & Yoveva, M. (1996). Context effects on problem solving. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*. Erlbaum, Hillsdale, NJ.

Koning, H. & Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B: Planning and Design*, 8:295-323.

Laird, J.E., Newell, A., & Rosenbloom, P.S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence*, 33:1-64.

Laird, J.E., Rosenbloom, P.S., & Newell, A. (1986) Chunking in SOAR: The anatomy of a general learning mechanism, *Machine Learning*, 1:11-46.

Lampinen, J, & Oja, E. (1992) Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision.* 2:261-272

Maes, P. (1991). *Designing Autonomous Agents*. MIT Press, Cambridge, MA.

McCarthy, J. (1990) Artificial Intelligence, Logic and Formalizing Common Sense. , Ablex, Norwood, New Jersey, 1990.

Marshall, J. B. (1999). *Metacat: A Self-Watching Cognitive Architecture for Analogy-Making and High-Level Perception.* PhD thesis, Indiana University, Bloomington, IN.

Mitchell, M. (1993). *Analogy-making as perception : a computer model.* MIT Press, Cambridge, MA

Mozer, M. C. (1994). Neural network music composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. *Connection Science, 6*:247-280.

Meyer, D.E. & Kieras, D.E. (1997). A computational theory of executive cognitive processes and multiple-task performance: I. Basic mechanisms. *Psychological Review*, 104(1):3-65.

Nelson, J. D., Cottrell, G. W., Movellan, J. R., & Sereno, M. I. (2004). Yarbus lives: a foveated exploration of how task influences saccadic eye movement. *Journal of Vision*, 4(8), 741a

Newell, A. (1980) Reasoning, problem solving and decision processes: The problem space as a fundamental category. In R. Nickerson (ed), *Attention and Performance VIII*. Lawrence Erlbaum, Hillsdale, NJ.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall, Englewood Cliffs, NJ.

Newell, A., & Simon, H. (1976). Computer Science as Empirical Inquiry: Symbols and Search. *Communications of the ACM*. 19(3):113-126.

Olshausen, B.A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607.

O'Reilly, R.C, & Munakata, Y. (2000) *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain.* MIT Press, Cambridge, Mass.

Pollack, J.B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77-105.

Pugliese, M. J. & Cagan, J. (2002). Capturing a rebel: modelling the Harley-Davidson brand through a motorcycle shape grammar. *Research in Engineering Design*, 13(3):139-156

Rehling, J. (2000). *Letter Spirit (part two): Modeling creativity in a visual domain.* PhD thesis, Indiana University, Bloomington, Indiana.

Rubin, E. (1915). *Visuell Wahrgenommene Figuren*. Gyldendalske Boghandel, Copenhagen.

Rumelhart, D. E., McClelland, J. L., et al. (1986). *Parallel Distributed Processing, vol*. 1. MIT Press, Cambridge, Mass.

Selfridge, O. G. (1955). Pattern recognition in modern computers. *Proceedings of the Western Joint Computer Conference*. Institute Electrical and Electronic Engineers, New York.

Simpson, G. B., & Kreuger, M. A. (1991). Selective access of homograph meanings in sentence context. *Journal of Memory & Language*. 30(6):627-643.

Stiny, G. & Mitchell, W.J. (1978). The Palladian Grammar. *Environment and Planning B: Planning and Design* 5:5-18.

Toppino, T. C. (2003). Reversible-figure perception: Mechanisms of intentional control. Perception and Psychophysics. 65(8):1285-1295.

Treisman, A., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*. 12: 97-136.

Tsuruta, N., Taniguchi, R, & Amamiya, M. (2000). Hypercolumn Model: A Combination of Hierarchical Self-Organizing maps and Neocognitron for Image Recognition. *Systems and Computers in Japan,* 31(2).

Tversky, A. & Kahneman, D. (1981). The Framing of Decisions and the Psychology of Choice. *Science*, 211:453-458.

Ullman, S. (1998). Three-dimensional object recognition based on the combination of views. *Cognition. Special Image-based object recognition in man, monkey and machine.* 67(1-2):21-44.

Vecera, S. P., & O'Reilly, R. (1998). Figure-Ground Organization and Object Recognition Processes: An Interactive Account. *Journal of Experimental Psychology: Human Perception and Performance*, 24(2):441-462.

Weisberg, R. W. (1988). Problem solving and creativity. In R. Sternberg (Ed.). *The nature of creativity* (pp. 148 – 176). Cambridge University Press, New York.

Yarbus, A. L. (1967). Eye movements during perception of complex objects, in L. A. Riggs, ed., *Eye Movements and Vision*, chapter VII, pp. 171-196. . Plenum Press, New York.

Appendix A
Running FAE

System Requirements

FAE will run under both Windows and Linux. Since the graphical interface is computationally expensive to generate, it is recommended that you use a system faster than 600MHz, with at least 256Mb RAM. You will also need at least 100MB free hard drive space. If you are running from Linux, you will need to have the Java Runtime Environment installed (at least v1.4.2) and included in your path.

Getting Started

The CD provided contains the program code for running the main simulations contained in this thesis. The program is contained in the folder called "FAE". Although the program will run directly from CD, it is recommended that you copy the "FAE" folder onto your hard drive.

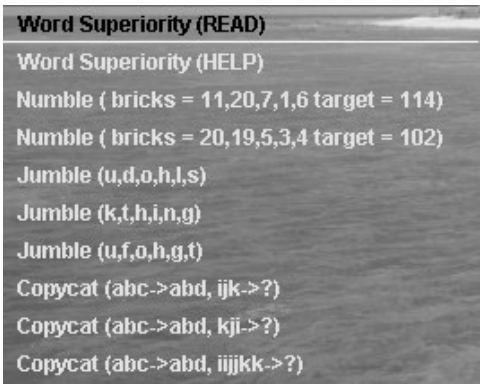
In order to run FAE from *windows*, use windows explorer to open the file in the FAE folder called "FAE.bat". In order to run the program under Linux, open up a terminal, cd into the FAE folder and run the file called "unixFAE" (by typing ./unixFAE) (note: you may have to change the permission to executable on this file by typing chmod +x unixFAE).

When you run the startup script, the following window will appear:

The screen is divided into three main sections: the workspace (the main area), a project tree (to the left) and a menu bar below. To load an example project, click on the file button on the menu bar to the bottom left of your screen. The following menu will appear:

To run an example simulation from this thesis, click on the “Load Example” option.

When you click on "Load Example" option, you will be provided with a list of example problems from the various projects:



Simply click on the example problem that you wish to load. After a few moments, FAE's working memory will appear in the workspace:

To run the loaded problem, simply click on the play button. If you are running problems from the Copycat domain, there may be several alternative solutions. In this case, after a solution has been found hitting stop to reset the contents of working memory and replaying the problem may yield different results.